NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A DESIGN METHODOLOGY FOR EMBEDDED WEAPONS SYSTEMS USING THE HARPOON SHIPBOARD COMMAND-LAUNCH CONTROL SET (HSCLCS), AN/SWG-1A(V)

by

Daniel P. Olivier

and

Kevin R. Olsen

Thesis Advisor: Ronald Modes

Approved for public release, distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A Design Methodology for Embedded Weapons Systems Using the Harpoon Shipboard Command-Launch Control Set (HSCLCS), AN/SWG-1A(V)	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, June 1983
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Daniel P. Olivier and Kevin R. Olsen	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: June 1983
	13. NUMBER OF PAGES: 115
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Harpoon Weapons System, Software Engineering, Software Design, Data Flow Diagrams, Program Design Language, Ada as a PDL

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis demonstrates a structured approach to software development for a complex weapons system. The project is the redesign of the Weapons Console Indicator Panel (WCIP) for the Harpoon Shipboard Command Launch Control Set (HSCLCS). A methodology is presented which takes the design from the requirements analysis phase, through data flow diagrams and transform analysis, up to the actual design in a System Design Language (SDL).

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Approved for public release, distribution unlimited

A Design Methodology for Embedded Weapons Systems Using the Harpoon Shipboard Command-Launch Control Set (HSCLCS), AN/SWG-1A(V)

by

Daniel P. Olivier
Lieutenant, United States Navy
B.S., United States Naval Academy, 1977

and

Kevin R. Olsen
Lieutenant, United States Navy
B.S., United States Naval Academy, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
June 1983

ABSTRACT

This thesis demonstrates a structured approach to software development for a complex weapons system. The project is the redesign of the Weapons Console Indicator Panel (WCIP) for the Harpoon Shipboard Command Launch Control Set (HSCLCS). A methodology is presented which takes the design from the requirements analysis phase, through data flow diagrams and transform analysis, up to the actual design in a System Design Language (SDL).

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Software development for complex weapons systems has proven to be extremely costly in both time and money. The Department of Defense is currently spending $4.8 billion a year on software, nine times the amount spent on hardware [Ref. 1]. A systematic approach to the design, coding, and implementation of software is needed if these substantial costs are to be controlled.

This thesis demonstrates a structured approach to software development for a complex weapons system. The project is the redesign of the Weapons Console Indicator Panel (WCIP) for the Harpoon Shipboard Command Launch Control Set (HSCLCS). A methodology is presented which takes the design from the requirements analysis phase up to the actual design in a System Design Language (SDL).

B. SOLUTIONS TO SOFTWARE PROBLEMS

The Department of Defense has recognized the escalating costs of software development and has initiated the design of a new programming language, Ada, as a major step in implementing a structured approach to software design. DOD acknowledges that Ada is not a panacea and is only one of many tools that are needed. Studies made by the Defense Advanced Research Projects Agency and Decisions and Designs,

Inc., predicted that the use of Ada as a standard programming language would save DOD $24 billion in software development costs from 1983 to 1999 [Ref. 2].

In addition to the use of ADA, new methods of designing software are currently being developed. Rodger Pressman advocates the use of Data Flow Diagrams for a design methodology, followed by transformation analysis which implements a control structure on the modules derived from the Data Flow Diagrams. Grady Booch has suggested the use of an Object Oriented Design that is well suited for the use of Ada. The main consideration remains that some well thought out design methodology must be adapted to reduce the overruns that are presently associated with software development.

C. A SOFTWARE LIFECYCLE APPROACH

In addition to cost and schedule overruns that are associated with software development, the cost of maintaining large programs once they are delivered is significant. Maintenance includes efforts to fix programs that are incorrect and to enhance existing code. Estimates of maintenance costs range from 50-90% of the total lifecycle cost of software [Ref. 3]. A design methodology must address the problem of maintainability of the final software product.

To enhance maintainability of software, emphasis must be placed on readability and modularity of programs during the software development process. A step in the reduction of

maintenance and development costs can be the creation of program libraries. This would allow the use of code generated for any one project to be used for the enhancement or creation of other software programs. Creation of large projects would be greatly simplified if they could call upon existing smaller programs. These benefits can only be fully realized if software is developed in a modular fashion. Software should not be viewed as having been developed solely for a particular application and then restricted to that use only. Software can be envisioned as a library of modules or packages that can be discarded and replaced if incorrect or no longer adequate and as a set of modular building blocks for the construction of other programs.

D. SUMMARY

This thesis is an attempt to demonstrate the use of a combination of design methodologies in the development of the Harpoon Weapons system. The initial design was presented in a thesis by Larry Sentman and Randy Maroney, "Integrated Design Specifications for the Harpoon Shipboard Command Launch Control Set." The authors of this thesis will refine the initial design and present a step by step process for designing a major system.

Through the combination of many different techniques, the authors will develop a unique logical approach to the

software engineering problem associated with complex embedded weapons systems.

Data Flow Diagrams will be the first step. This approach allows the data or information to be charted throughout the system. A Transform Analysis is then performed on the Data Flow Diagrams. This analysis imposes a control structure on the data and a hierarchical modular design is created. Once modules are identified, modules can then be grouped into packages to be used by Ada for a System Design Language (SDL) or a Program Design Language (PDL).

II. HARPOON SHIPBOARD COMMAND-LAUNCH CONTROL SET (HSCLCS) AN/SWG-1A(V) SPECIFICATIONS

The purpose of this chapter is to summarize the initial system specifications as presented in Ref. 4. This is the first phase of the software engineering process and addresses the following: definition of existing system, statement of the needs of the existing system, and a statement of the technical constraints imposed by hardware considerations.

A. EXISTING HARPOON WEAPON SYSTEM

The Harpoon Weapon System (HWS) has been developed to fulfill the requirements of the Navy's anti-ship mission. The HWS is currently deployed on surface combatants, fast attack submarines, and an assortment of aircraft. The HWS is to provide an anti-ship capability, at over the horizon ranges in an all-weather, day or night environment.

The HWS is comprised of the missile subsystem, the associated launcher subsystem, and the command and launch control subsystem.

The Harpoon missile employs a low-level trajectory during flight with a pop-up maneuver in the terminal phase. It contains an active radar seeker head with counter-counter measures to attack surface targets at over-the-horizon ranges. The missile is essentially identical for ship,

submarine and aircraft launch. The significant difference is a booster is added for ship and submarine launch.

The ship-launched HARPOON employs either onboard or third party sensor data for targeting information. The missile is a "launch and forget" weapon, since no ship control or information is needed after launch.

For surface ships, the HWS control and data processing functions are provided by the Harpoon Shipboard Command-Launch Control Set which has three modes of operation: normal, casualty, and training. In the normal mode the major functions provided by the HCSLCS include:

- Distribution of power to various HWS equipment.
- Selection and application of missile warmup power.
- The ability to conduct various automatic and manually initiated tests which confirm the operability of the HSCLCS.
- Distribution of ship motion and speed data from ship equipment.
- Selection, transfer, processing and display of target data.
- Coordination of the selection of tactical missile mode and type of fusing.
- Selection of the launcher cell containing the intended HARPOON missile.
- Initialization of the selected missile and the supervision of the exchange of data between the missile and other HWS equipment.
- Control of all missile firing activities.

These functions are implemented and integrated by the HARPOON Weapon Control Console (HWCC) and the Weapon Control Indicator Panel (WCIP).

The HWCC contains most of the HARPOON system-unique command and launch subsystem equipment, including the Data Processor Computer (DPC), the Data Conversion Unit (DCU) and the HWCC life support equipment. Together these HWCC components perform data processing and conversion among various data types and provide interfacing with existing sensor and ship's equipment.

The WCIP provides visual status information to the operator during formulation of the fire control problem, and additionally provides manual controls for the operator. The existing WCIP is shown in Figure 2-1.

The DPC is a 16-bit microcomputer with 15K of Eraseable Programmable Read Only Memory (EPROM). The DPC uses an assembly language program to provide the following functions:

- Launch envelope parameter validation.
- Missile command generation for implementation of missile control parameters including ship's attitude, search pattern orders, engine starting, flight termination range, altimeter setting, and various selectable flight trajectory and maneuvering modes.
- Pre-launch testing of the missile.
- Pre-launch sequencing and timing.
- Data formatting and transfer synchronization.

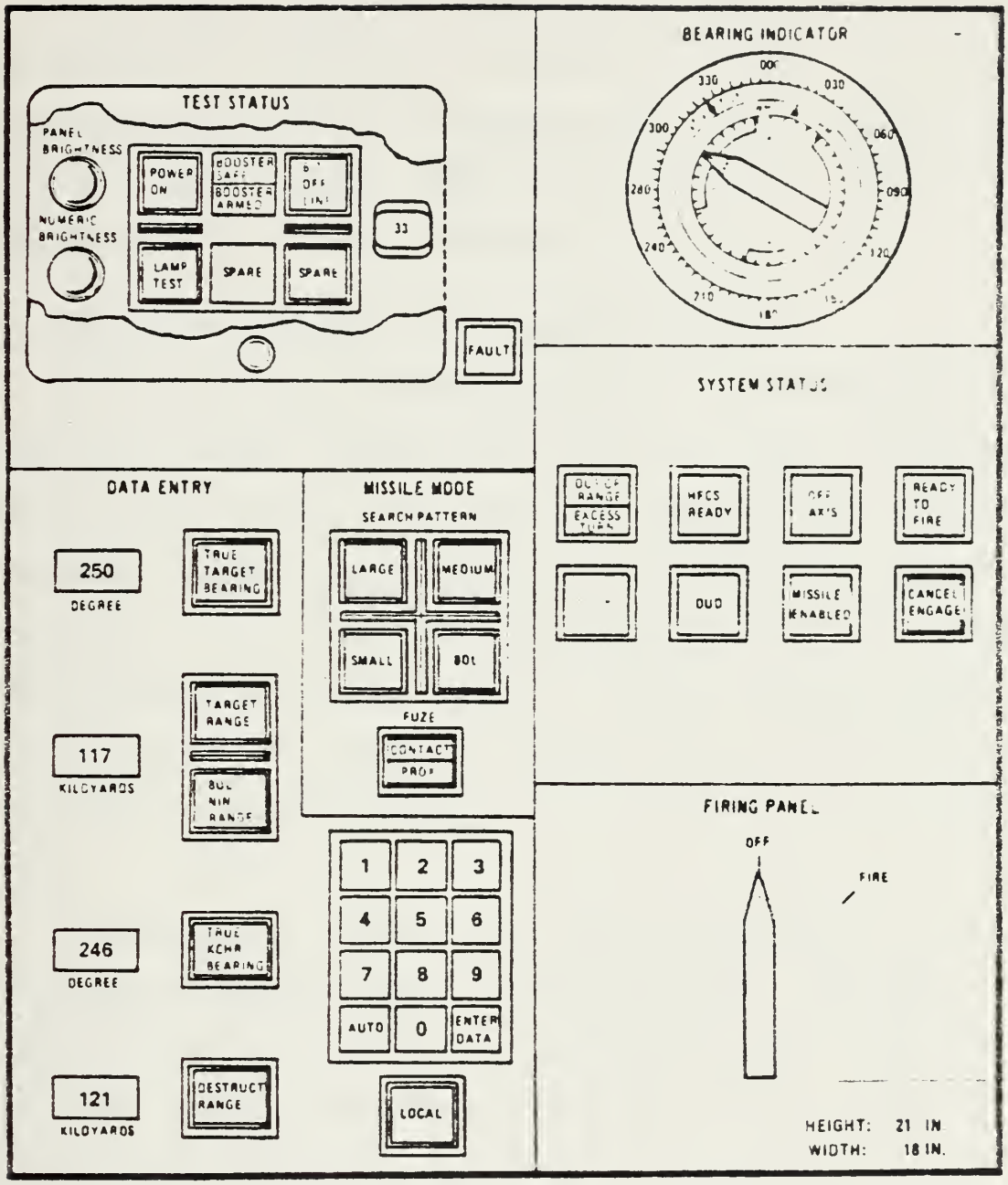


Figure 2-1 Representative Cannister Launch System WCIP

The DCU processes all digital and analog signal conversions as required by installed hardware. The DCU also provides interfacing of target data inputs from the Naval Tactical Data System (NTDS) Slow Interface. Ship motion parameter data is also converted in the DCU.

B. PROBLEMS ASSOCIATED WITH EXISTING HSCLCS

Successive block enhancements in the HARPOON missile have introduced added command and control problems. The existing WCIP cannot handle the improved capabilities of the new warhead (Block 1C). With the present WCIP's buttons and display, the operator is ill-equipped to direct and execute a well-formulated Harpoon attack. The new and projected capabilities of the missile cannot be fully utilized without substantial hardware and software modification within the WCIP.

The existing software for the present HSCLCS is written in machine language and is heavily hardware dependent. These reasons lead to a relatively high maintenance cost and make changes to the software difficult. Considering that different hardware configurations exist for surface, subsurface and air launches, this hardware dependence only compounds the difficulties of standardizing the software.

The Maroney-Sentman thesis has identified the following HSCLCS deficiencies; with regard to engagement planning:

- Full tactical control of existing missile variants (the pre-launch selections) are not available to the existing WCIP.

- The WCIP provides inadequate control for a well coordinated, multi-ship or multi-platform attack against a single surface target.

- The WCIP provides inadequate control for a multi-missile (SALVO) attack against a single surface target.

- The WCIP does not incorporate existing intelligence information (e.g., target class, course, speed, sector of vulnerability) into the engagement planning process.
- No computer-aided engaged planning is implemented.

With regard to the analysis of the engagement plan:

- Insufficient information is displayed at the WCIP to permit the operator to evaluate the quality of an engagement plan (e.g., probability of acquisition).

- Insufficient information is displayed at the WCIP to provide accurate data implying risk to unintended targets during booster drop, flyout and target acquisition.

- The WCIP provides no display of planned trajectory, flight path or seeker search patterns.

- The HSCLCS does not provide computer-aided engagement plan quality and safety analysis.

- The WCIP provides no status information on available missiles and associated launcher.

At the present time only track data for one track can be stored, with no provisions for multi-track data retention.

Environmental parameters such as wind, rain, and sea state influence missile performance. No means are currently available to input this information or to provide corrections essential to missile performance.

C. HARPOON WEAPON SYSTEM CONSTRAINTS

The upgrade for the HSCLCS must be able to take full advantage of the new Block 1C missile capabilities, but must also remain compatible with Block 1A and Block 1B. Although the Block 1C is currently being manufactured, the 1A and 1B will continue to be operationally deployed throughout their normal service life.

The implementation of the upgrade to the HSCLCS must continue to provide all previous pre-launch functions. In addition the upgrade must maintain interface compatability with the Naval Tactical Data System (NTDS) Slow Interface.

The existing launcher hardware including launcher control and test equipment will not be subject to change for the upgraded HSCLCS.

Because the HSCLCS is installed on various surface platforms, where space is limited in many cases, the physical size of the HSCLCS must remain the same.

The Built-In-Test (BIT) and Built-In-Test equipment (BITE) requirements established for the existing HSCLCS will remain effective for the upgrade HSCLCS.

While the DCU hardware configuration must remain the same, the DPC software is subject to change as necessary to implement the upgraded HSCLCS. As discussed previously, present software is written in machine language, software

changes will be both difficult and expensive to develop, test and maintain.

System reliability, hardware maintainability and system environmental standards for the HSCLCS upgrade must meet or exceed the performance specified for the existing HSCLCS.

D. SYSTEM DEFINITION FOR HSCLCS UPGRADE

1. Introduction

The purpose of this section is to give a concise, general overview of the hardware and software description for the HSCLCS as presented by the Maroney-Sentman thesis.

a. System Objectives

The prime objective of the HSCLCS upgrade is to provide for the full tactical deployment of all missile options associated with the Block 1C HARPOON missile.

The tactical options introduced by successive block enhancements has not been sufficiently addressed from the perspective of operator control. A considerable improvement in the amount of positive operator control during a tactical employment of the HWS is a system design objective.

At present, no graphical display representing the local tactical surface warfare scene has been directly available to the HARPOON operator. A tactical display will improve operator comprehension of the tactical situation and

assist in planning and execution. An improvement in tactical employment is the objective.

Historically, the introduction of block enhancements to the HARPOON missile has required WCIP hardware modifications. The use of programmable function keys for operator control may alleviate this requirement.

Another objective of the HSCLCS upgrade is to assist the operator in engagement planning and analysis. Automatic calculation and display of probabilities of acquisition is a valuable aid for the measurement of the relative quality of a planned engagement prior to expending missiles.

b. Hardware Component Overview

Figure 2-2 is an overview of the HWS. The only item of hardware to be changed is the WCIP and its attached display console.

2. System Hardware Functional Description and Allocation

The HSCLCS upgrade requires only changes in the HSCLCS subsystem. The missile and launcher subsystems remain intact.

a. HSCLCS Subsystem

The HSCLCS upgrade requires modifications to both the HSCLCS hardware and software components.

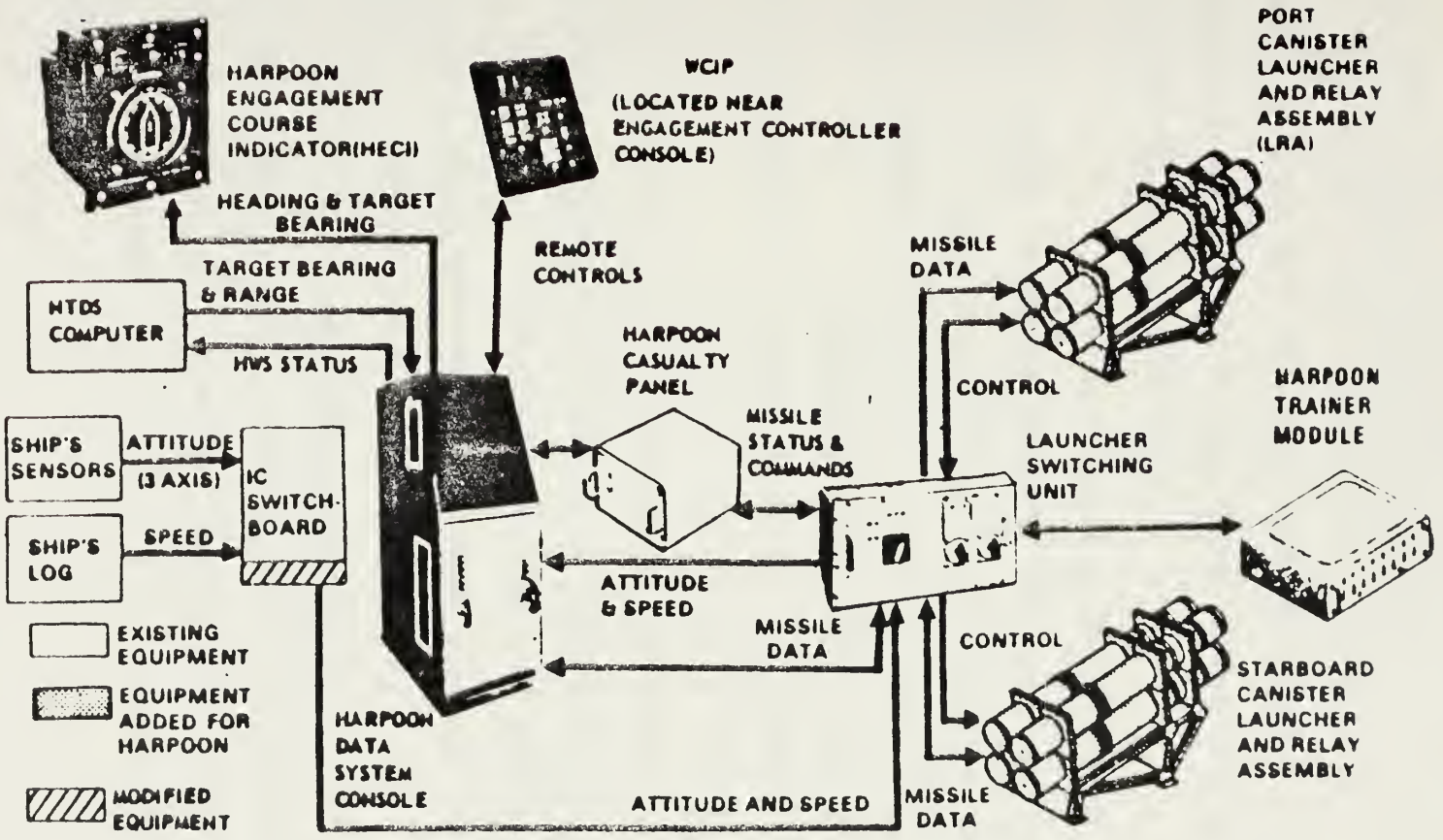


Figure 2-2 Hardware Component Overview of the Cannister Launch HARPOON Weapon System

(1) Weapons Control Indicator Panel. A new display and operator console is planned for installation into the existing WCIP enclosure. Figure 2-3 depicts the preliminary design mockup of the new WCIP. Figure 2-4 is an enlarged view of the display area of the replacement WCIP and the associated function keys.

The new display provides a full tactical display using a plasma display panel instead of conventional cathode ray tube. An attached microprocessor will process all screen graphics software routines as commanded by the DPC.

Programmable software keys for operator use will be located on either side of the display screen.

A [hook] and [break] button along with a cursor control will be mounted on the right side of the WCIP. This arrangement is similar to a standard NTDS console.

A firing key, a set of missile and launcher states lights, and miscellaneous display console controls, are all included in the new WCIP.

(2) Data Processing Computer. The existing DPC microprocessor will be replaced with a commercially available CPU with additional memory. New software, with the exception of display graphic software, will be processed by the new DPC.

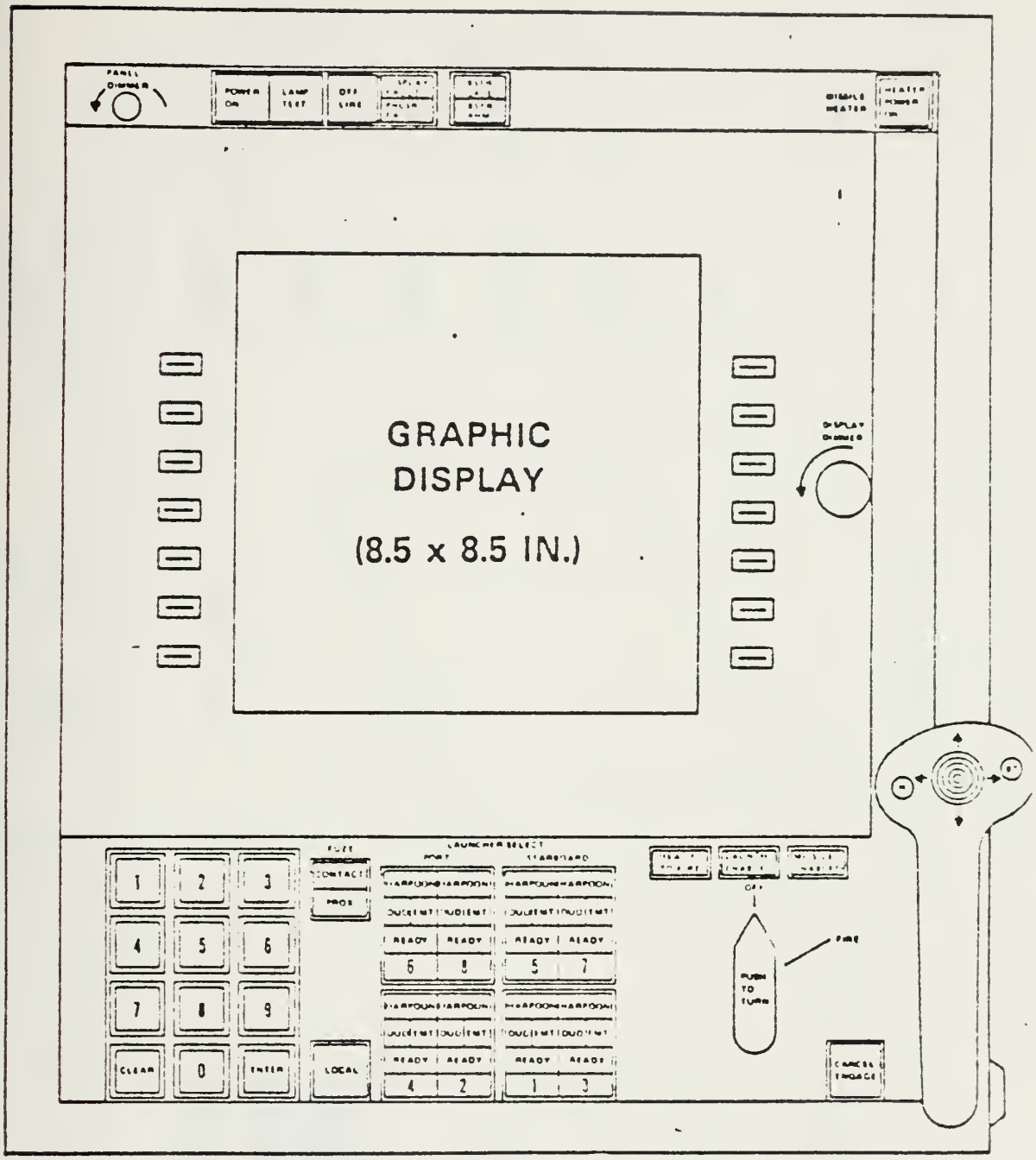


Figure 2-3 Proposed WCIP

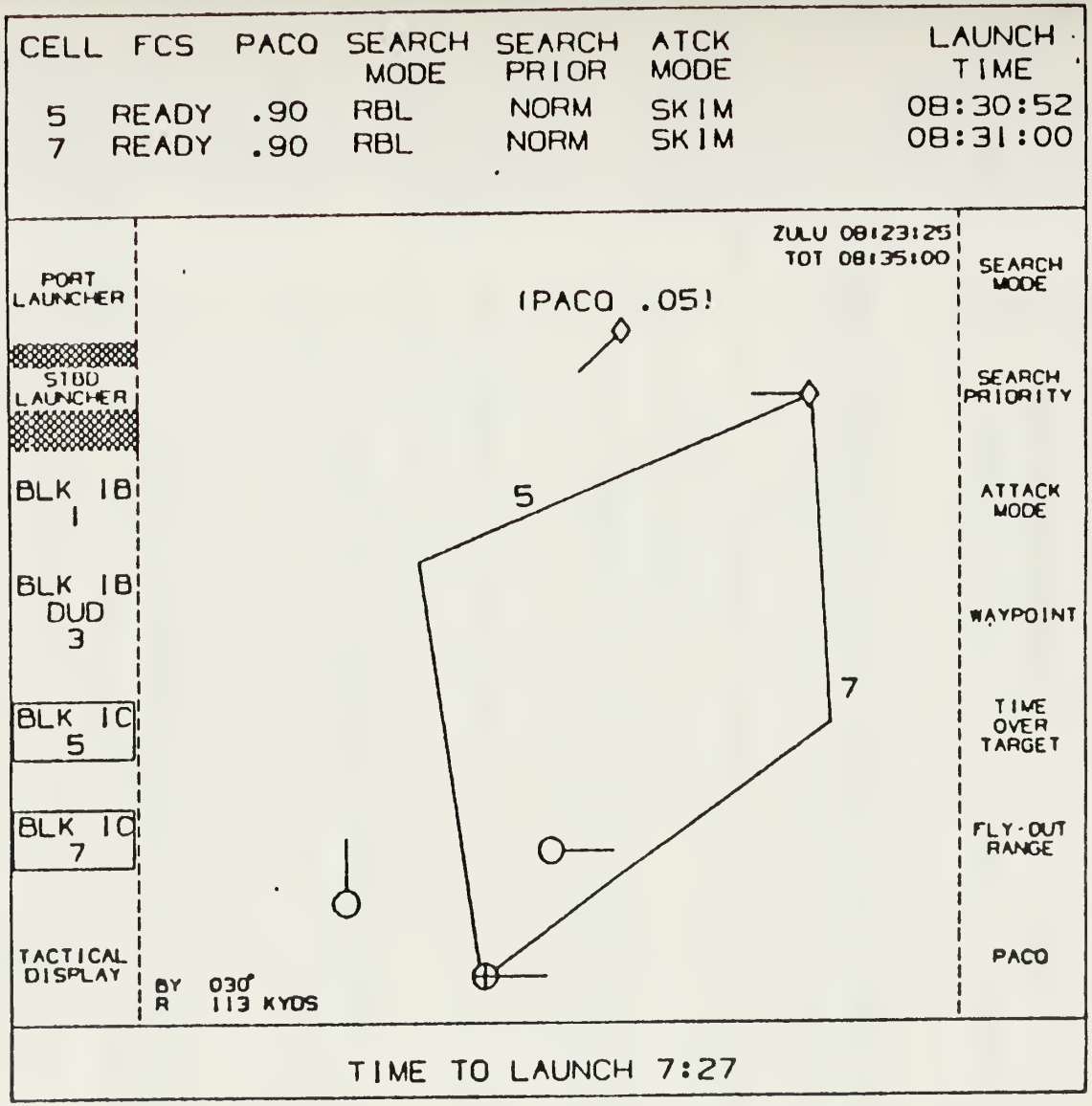


Figure 2-4 Display Area of Replacement WCIP

(3) Data Conversion Unit. No change in DCU hardware is permitted. DCU software changes are permissable only to interface with the data sources providing input for new DPC processing requirements.

(4) Display Processor. The display graphics software will be processed by an attached processor in the WCIP.

3. System Software Functional Description and Allocation

Graphics display software will not be addressed by the authors. Remaining HSCLCS software shall be processed by the DPC. A general discussion follows.

a. General HSCLCS Software System Specifications

General purpose HSCLCS software includes essential interfacing, interprocess communications protocol, and state transition management.

(1) Interfacing Software Specification. Detailed software requirements for interfacing will be addressed in Chapter V.

(2) Software Support of Existing Missiles. Any new HSCLCS software must remain operable with all USN missile subsystem variants through Block 1C missile variants.

(3) Software Support of Existing Launchers. HSCLCS software is required to provide launcher support functions for all existing launcher configurations. See Ref. 4 for a detailed review.

(4) State Transition Management Software. Long-term functional utility of the WCIP display console is insured by use of programmable function keys. To implement this architecture the following are required:

- Provide display button labels for each operator control function. These labels must be organized into logical sets, or menus, which will be displayed as a unit. The menus correspond, one-to-one, with a given overall system control state.

- Implement a state transition matrix to provide a mapping from a given state to a corresponding menu, with individual button meaning uniquely defined for a given state.

- Provide for the sequencing of events necessary to implement a state transition. When a control command is received, decode the command. If a state transition command is invoked, the change in control state shall be recorded and a new menu sent to the display screen. A critical period, when no commands are to be read, exists during the actual transition sequence.

- Provide for the decoding of all state-dependent inputs.

b. Operator Control Interface Software

The WCIP is the central point of control for the HWS. The WCIP provides the operator mechanisms for input of control commands and data.

(1) Display Output Software. The HSCLCS control related display functions are as follows:

- Display programmable button labels indicating HSCLCS operator functional choices in a reserved screen area adjacent to the corresponding function button.

- Provide for the operator queues and messages in a specifically reserved screen area.

- Display illegal action alert.

- Display a notice of lockout of any operator selected action.

- Display ZULU or local time as selected by the operator. The default time is ZULU.

- Display cursor position by a symbol such as a small circle similar to NTDS cursor display.

The display is the only form of feedback available to the operator during the engagement planning process. Engagement related display functions are as follows:

- Display options as selected for each engagement plan.

- Display projected flight path for a planned or partially planned engagement.

- Display time for launch for a planned attack.

- Display projected time of impact for a missile in flight.

- Display time desired for impact when a coordinated, multi-platform attack is selected.

- Display a data age alert for engagements using targeting data exceeding maximum age limitations.

- Display launch inhibit alerts and the respective cause.

- Display a notice that the flight path, as planned, exceeds the maximum range of the missile variant selected.

- Display environmental parameters as they are set by the operator or as requested.

- Display land mass representations as entered by the operator.

- Display the booster drop zone projected for a given missile.

- Display a graphic representation of waypoints when selected for an engagement.

- Display minimum and destruct ranges when selected for an engagement.
- Display a graphic representation of search pattern expansion when selected for an engagement.
- Display a graphic representation of active radar seeker search area for an engagement.
- Display the point of descent with a marker when high attitude fly-out is selected.
- Display the off-axis turn angle numerically in degrees for a selected aimpoint.
- Display the selected terminal attack mode.
- Display the probability of acquisition for an intended target.
- Display missile ready notices.
- Display missile launch progress reports including cell or rail empty or missile dud.
- Display least capable missile variant which will perform the designate engagement profile.
- Display missile resource data including variant identifier, individual missile status (if other than fully operational), and cell or launcher location.

Track related display software functions are central to the idea of improved tactical comprehension by the operator. Track related display functions are as follows:

- Display own ship's position with standard NTDS symbology.
- Display surface tracks with standard NTDS symbology.
- Display air tracks with standard NTDS symbology.
- Display true course leaders of fixed length for surface contacts with a known course.
- Graphically distinguish an operator designated target.

- Display true bearing and range to a designated target.
- Display a notice of failure to correlate targeting data.
- Display a line of bearing as input manually by the operator.

c. Track Data Base Maintenance System

The Track Data Base Maintenance System (TDBMS) software provides all track data processing for the HSCLCS. The software must permit the receipt of targeting data from manual input, NTDS, own ship's sensors and third party sensors. This raw data must then be converted into a common reference for correlation. The track data base system then uses this data to maintain a current data base representing the tactical surface picture.

(1) Track Data Base Maintenance Software Functions. The following are data base maintenance functions:

- TDBMS software must provide for the initialization of a track record for both surface and air contacts as required by explicit "new track" notification.
- TDBMS software shall maintain the own ships track record based on dead reckoning of ship's position and motion data.
- TDBMS software shall remove a track designated for deletion from the data base.
- TDBMS software shall be capable of deleting a designated track from data maintenance, and also capable of restoring it upon command.
- TDBMS software shall provide for rapid access to an existing track by any user of the track data.

- Write access protection by mutual exclusion of competing processes shall be provided on the track record level.

- Read access to a track record is unrestricted.

- Track records shall contain at the minimum, the track position in the normalized track coordinates, track unique identifier, sensor source type identifier, track size (small or large), targeting data quality indicator value, track history headed by last known course and speed, time stamp indicating the time of the most recent report, track classification identifier (i.e., hostile, friendly, or unknown), absolute track identifier by ship class or unit name, true bearing and range from own ship, a time stamp and a linkage pointer to establish engagement plan where one exists for a particular track.

d. Engagement Planning System Software

The Engagement Planning System (EPS) is a software system whose purpose is to coordinate the formulation of an engagement plan for a designated target.

(1) General Engagement Planning Software Functions. The following are general EPS software functions:

- Support engagement planning for all missile variants through Block 1C.

- Respond to all manual and NTDS engagement related orders.

- Provide for the control and use of all tactical missile selectables.

- Be capable of computing the projected time of occurance of key events of a planned engagement.

- Be capable of calculating missile attack boundaries.

- Be capable of reading specific and non-specific track records from the Track Data Base.

- Be capable of reading specific and non-specific resource data from the Missile Resource Data Base.

- Be capable of reading environmental data.

- Be capable of reading ship's motion data.
- Record a finalized engagement plan in the Engagement Plan Data Base.
- Provide manual override for any portion of an autonomous engagement.
- Calculate the project flight trajectory.
- Submit a completed engagement plan to the Engagement Analysis System for engagement analysis.

(2) Manual Engagement Planning Software Functions. When in the engagement mode, EPS software shall provide for the logical and orderly selection of all missile employment options. As tactical variables are selected, they are recorded and displayed. A given selection may determine another set of logical options to be presented to the operator.

(3) Automatic Engagement Planning Software Functions. The provision for autonomous engagement planning is an objective for the HSCLCS upgrade.

At minimum, autonomous engagement planning software shall be capable of selecting the missile terminal mode based on known target size, selection of the fly-out mode, selection range and attitude required to clear shipping obstructions and the selection of the missile variant with the least performance options which is still capable of executing the engagement plan.

e. Engagement Plan Analysis Software Functions

The analysis of engagement plans is a stated objective of the HSCLCS upgrade. Each planned engagement shall be submitted to the Engagement Analysis System (EAS) for evaluation prior to launch.

III. SOFTWARE PLAN

A. INTRODUCTION

This chapter is the first refinement of the basic software plan as presented in the Sentman-Maroney thesis [Ref. 4]. For a first interpretation of the software plan see Ref. 4.

The requirements analysis is the first step in the planning phase of the software engineering development and should fulfill the following objectives:

- Provide a foundation for the software development by uncovering the flow and structure of information.
- Describe the software by identifying interface details providing an indepth description of functions, determining design constraints, and defining software validation requirements.
- Establish and maintain communication with the user-requester and the developer so that the preceeding two objectives may be satisfied.

B. AREAS OF REQUIREMENTS ANALYSIS

1. Problem Recognition

Problem recognition requires review of the software specifications and the software plan.

2. Evaluation and Synthesis

Evaluation and synthesis is the major effort during the software requirements phase. The flow of data and its structure, detailed refinement of the software functions and

discovery of design constraints are the steps to accomplish this portion of the design process.

C. DATA FLOW DIAGRAM (DFD)

The data flow diagram (DFD) is a graphical aid for showing the data flow of the software system being designed. A complete understanding of the DFD is very important to the understanding of the software engineering design method. The following is a synopsis of the use of the DFD.

1. DFD Attributes

- Information flow in any system can be represented by a DFD.
- Each bubble or transformation in any DFD may require significant refinement to establish complete understanding.
- Emphasize data flow. Do not worry about control of the data.

2. DFD Symbols

- Information flow is represented by a labeled straight line from the source to the sink with the arrowhead pointing to the sink.
- A process data transformation is represented by a circle called a bubble.
- Information sources and sinks are represented by rectangles.
- Stored information (e.g., data bases or files) are represented by two lines in parallel.

3. DFD Usage Guidelines

- The first layer of the DFD is always the system module.
- The second layer of the DFD should be the generalized or overview DFD.

- All arrows, bubbles, sources, sinks, etc. must have labels.

- Information continuity is required on DFD refinements. All incoming and outgoing arrows in the DFD being refined must appear in the refinement.

- Refine only one bubble at a time. The bubbles in the overview DFD are numbered with a single integer beginning at '1'. Then as they are subsequently refined, the expansion's numbers are added to by a '.' and another integer beginning at '1'. This numbering system is continued for all DFD's.

- Bubble refinement can yield bubbles, rectangles or two parallel lines in any combination.

- DFD's allow isolation of any domain of change.

- When there is uncertainty whether the DFD development is complete, assume that further refinement is possible and continue with the DFD refinement process.

- Follow data flow as a single thread from left to right. The DFD development may require a loop back to a previously defined transformation. Provisions for the single thread data flow where such a loop is required are made by duplicating the transformation so the flow continues from left to right.

- A transformation may output control data for a subordinate module. This control data does not represent control structure and therefore is not control flow.

D. HSCLCS DATA FLOW DIAGRAMS

Figures 3-1 through 3-10 represent the development of the HSCLCS system by the data flow method described in the section above.

The fundamental HSCLCS DFD is shown in Figure 3-1. The HSCLCS bubble is the domain of change that will be developed in the subsequent DFD's.

The first refinement DFD of the HSCLCS is shown in Figure 3-2. The HSCLCS is broken down into four (4) major bubbles which comprise the flow of information within the HSCLCS. These four bubbles are derived from the data flow analysis and are numbered to aid in subsequent refinements. These transforms constitute the heart of the new HSCLCS and have the following basic functions:

- Transform 1, Process Input, receives and processes all manual and automated inputs and transforms the data so that it may be properly routed to the correct data base for update, or passed to one of the other three transforms. This transform represents the input side of the WCIP, while Transform 4, Display Output, represents the transformation of the outputs to the screen display, and all other visual displays that are a part of the new WCIP.

- Transform 2, Update Track Data Base, processes both the manual input of Transform 1 and NTDS track data. The track data base is then used by Transforms 3 and 4.

- Transform 3, Plan Engagement, develops and sends launcher/missile orders when it receives the orders from the operator through Transform 1. The most complex algorithm is contained in this transform, that of determining an automatic engagement solution with the given input target information. Straight shots using a Block 1A/1B missile may only need a simple engagement algorithm. Complexities arise if waypoints are required, and even more complexities if waypoints are determined automatically by this transform.

- Transform 4, Display Output, takes all the data from the data bases maintained by the various transforms and operator manual display orders and then provides the transformation required for proper display.

The complete development of the DFD for the 1, Process Input transform, is shown in Figure 3-3. This bubble

identifies the transaction and passes the data to the proper data base for update, or to the proper receiving transform.

Figure 3-4 is the first refinement of the track data base DFD and leads to six new transforms. At this level, no distinction is made regarding the source of the track data. Two sources are possible, manual track data or NTDS track data.

Figure 3-5 is the complete refinement of the track data base DFD. Five additional transforms are derived, all of which update various portions of the track data base.

Figure 3-6 is the first refinement of Transform 3, Plan Engagement. This bubble is the most detailed DFD and also the most complex. Transform 3.1 and 3.3 provide interfaces between launcher missile status data base and manual engagement orders. This data is time stamped so that its age can be judged by those modules which use this information. Transform 3.2, Plan Engagement, develops an engagement solution upon operator request. The completed engagement plan is then passed to the engagement data base. This information is then available to Transform 3.1, Launcher Missile Assignment.

Figure 3-7 is a further refinement of Transform 3.2, Plan Engagement. This bubble uses the threat, launcher missile status, track, environmental, and ship parameter data bases to develop the actual engagement solution. The

uncertainty ellipse and probability of acquisition are also calculated and the finalized engagement plan is then placed in the engagement data base.

Expansion of 4, Display Output, leads to eight new transforms shown in Figure 3-8. Transform 4.1, Select Function, is driven by the manual display order. Transforms 4.2 through 4.7 all send display orders to Transform 4.8, Console Display, which insures a proper interface with the WCIP.

Transform 4.4, Display Engagement, is further broken down into 4.4.1-4.4.3, Threat Engagement and Graphics Display.

E. DATA STRUCTURE REPRESENTATION

The data structure of the major data bases of the HSCLCS design are detailed in the Data Structure Definition and are presented in Appendix B. These Data Structure Definitions detail the first-cut description of the data bases.

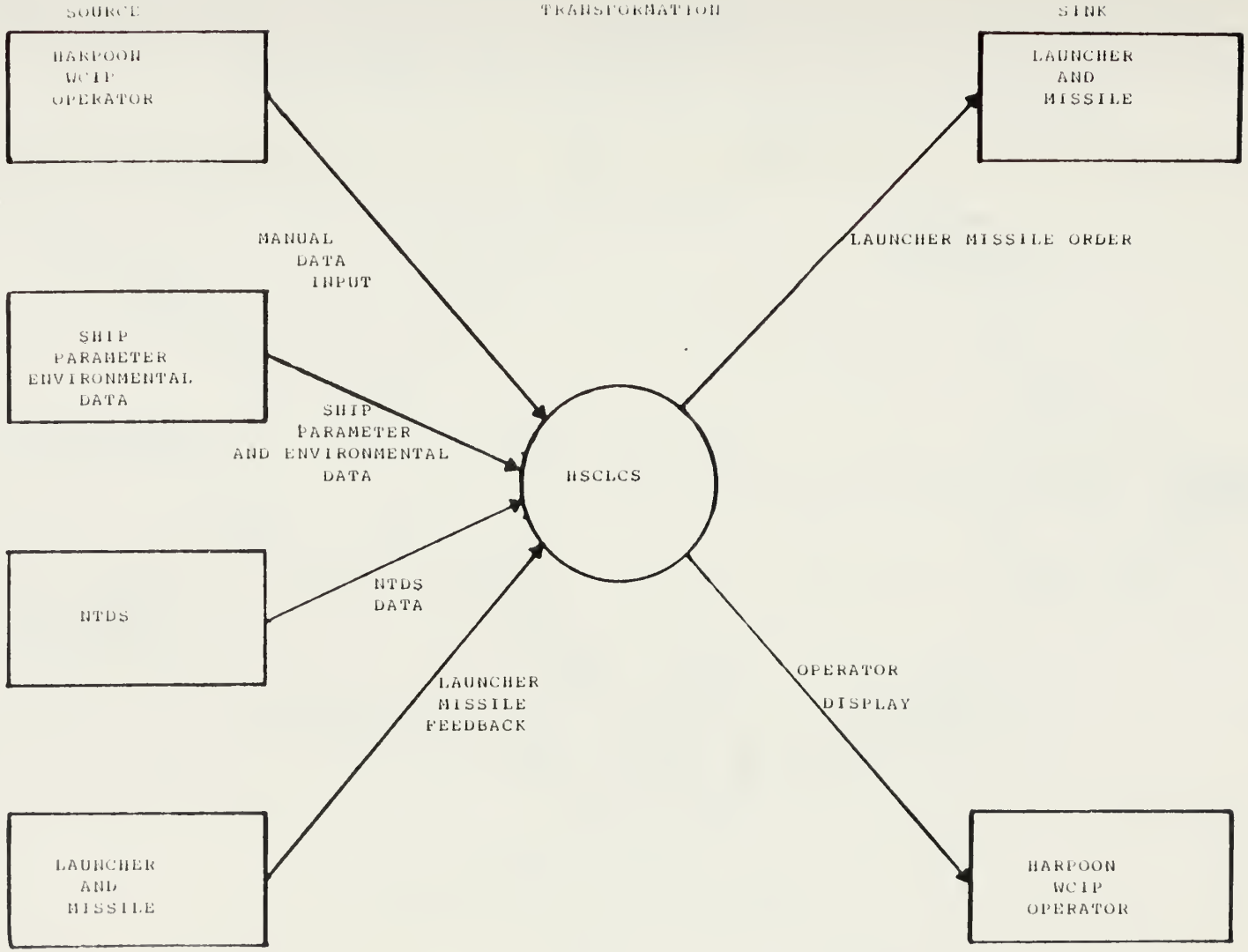


FIGURE 3-1
SOURCE/SINK DIAGRAM

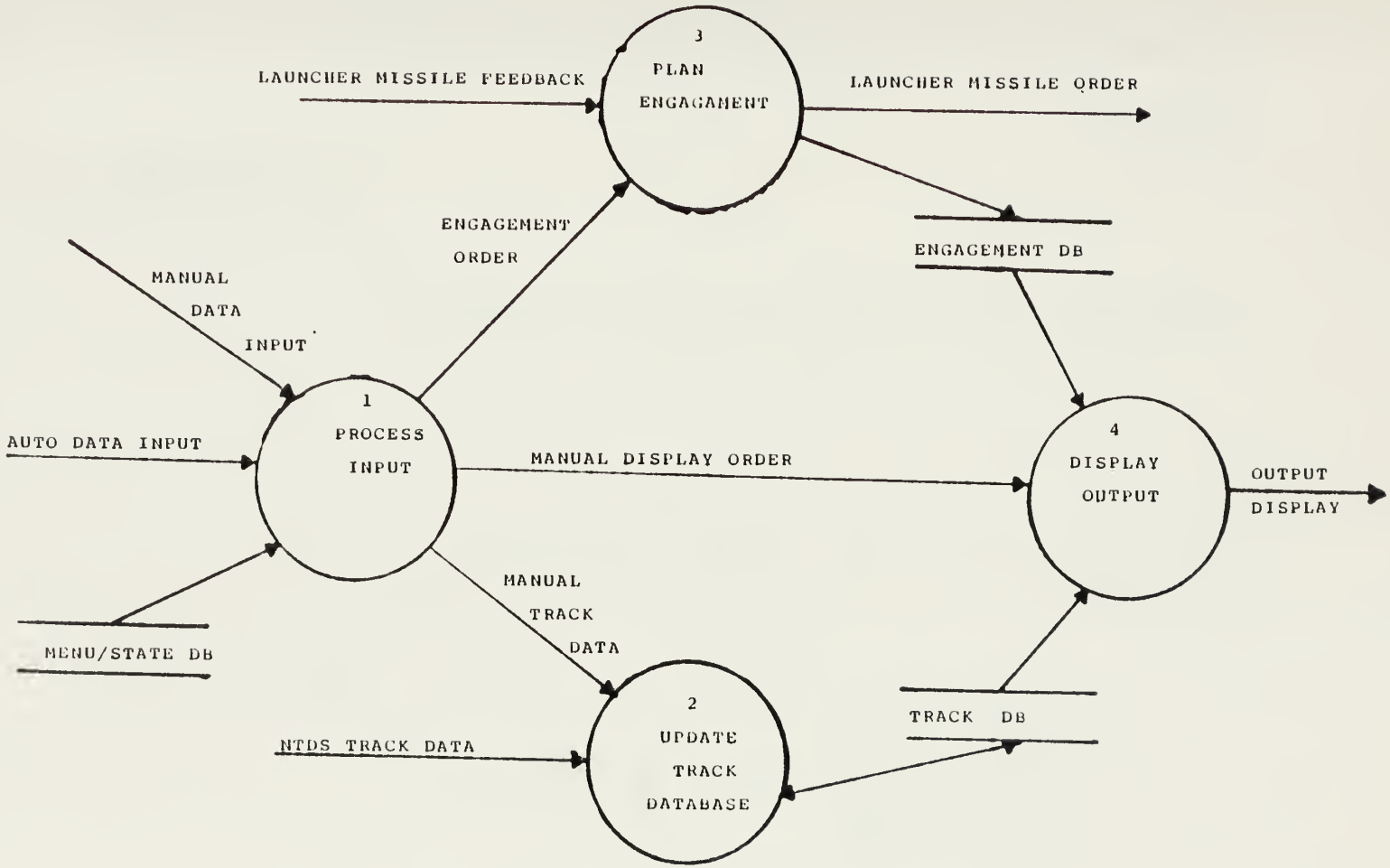


FIGURE 3-2

SYSTEM OVERVIEW DATA FLOW DIAGRAM

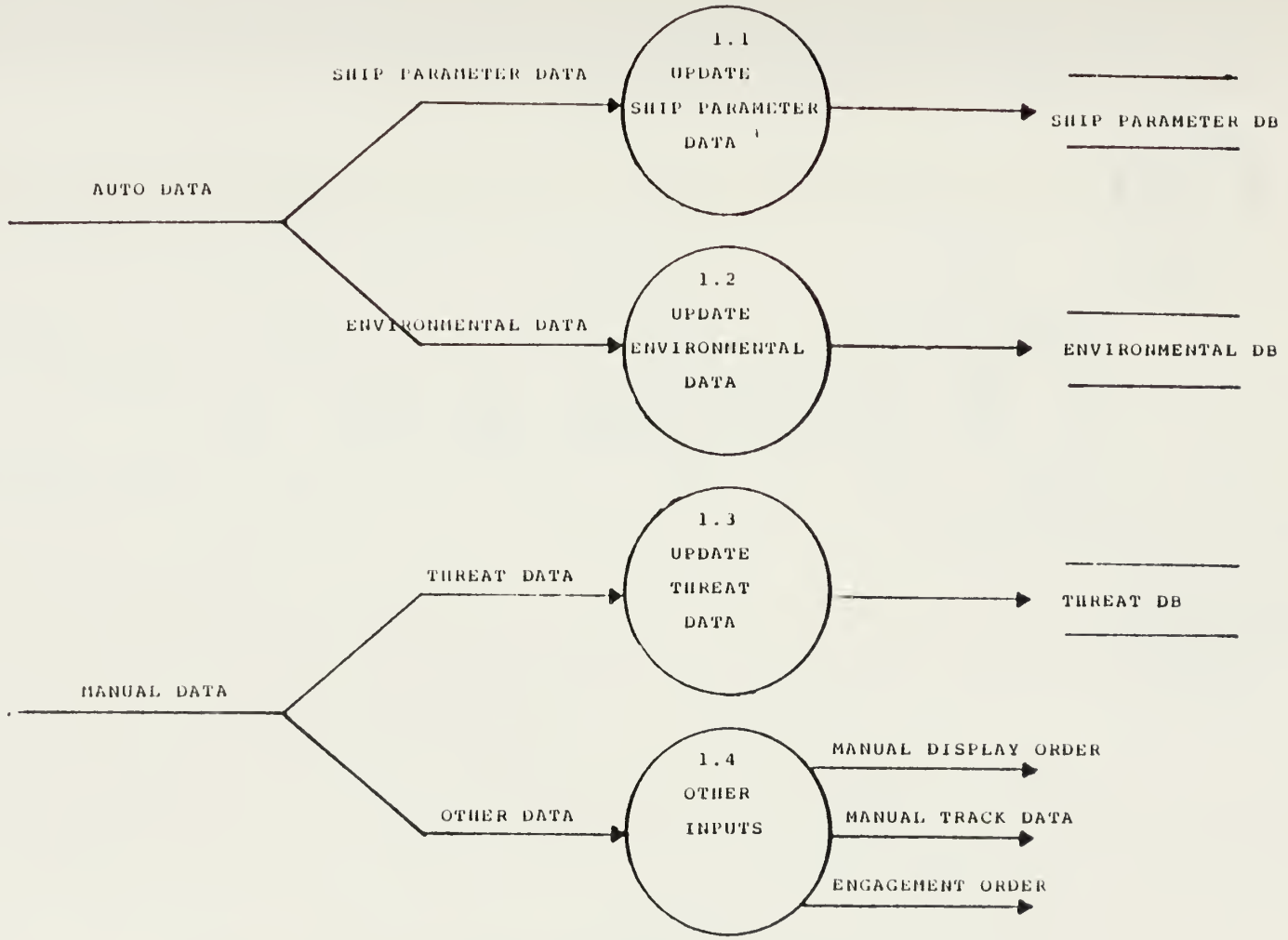


FIGURE 3-3

DECOMPOSITION OF PROCESS INPUT BUBBLE

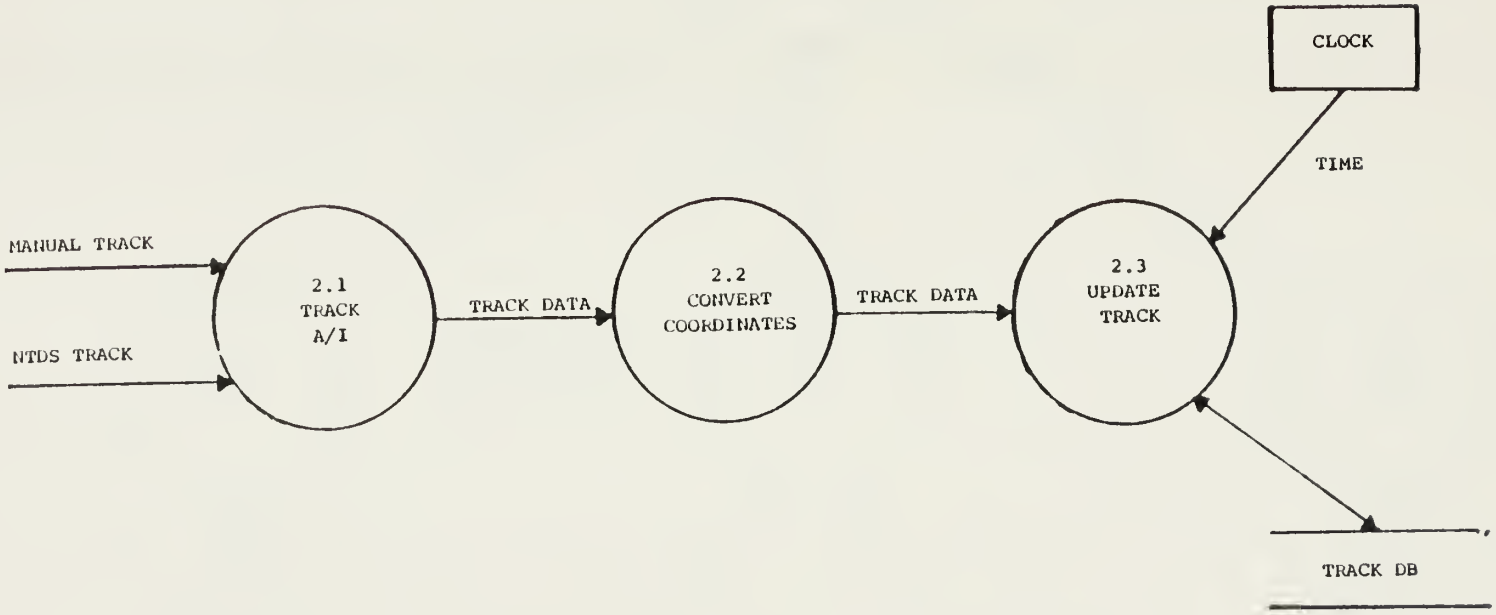


FIGURE 3-4

DECOMPOSITION OF UPDATE TRACK DATABASE

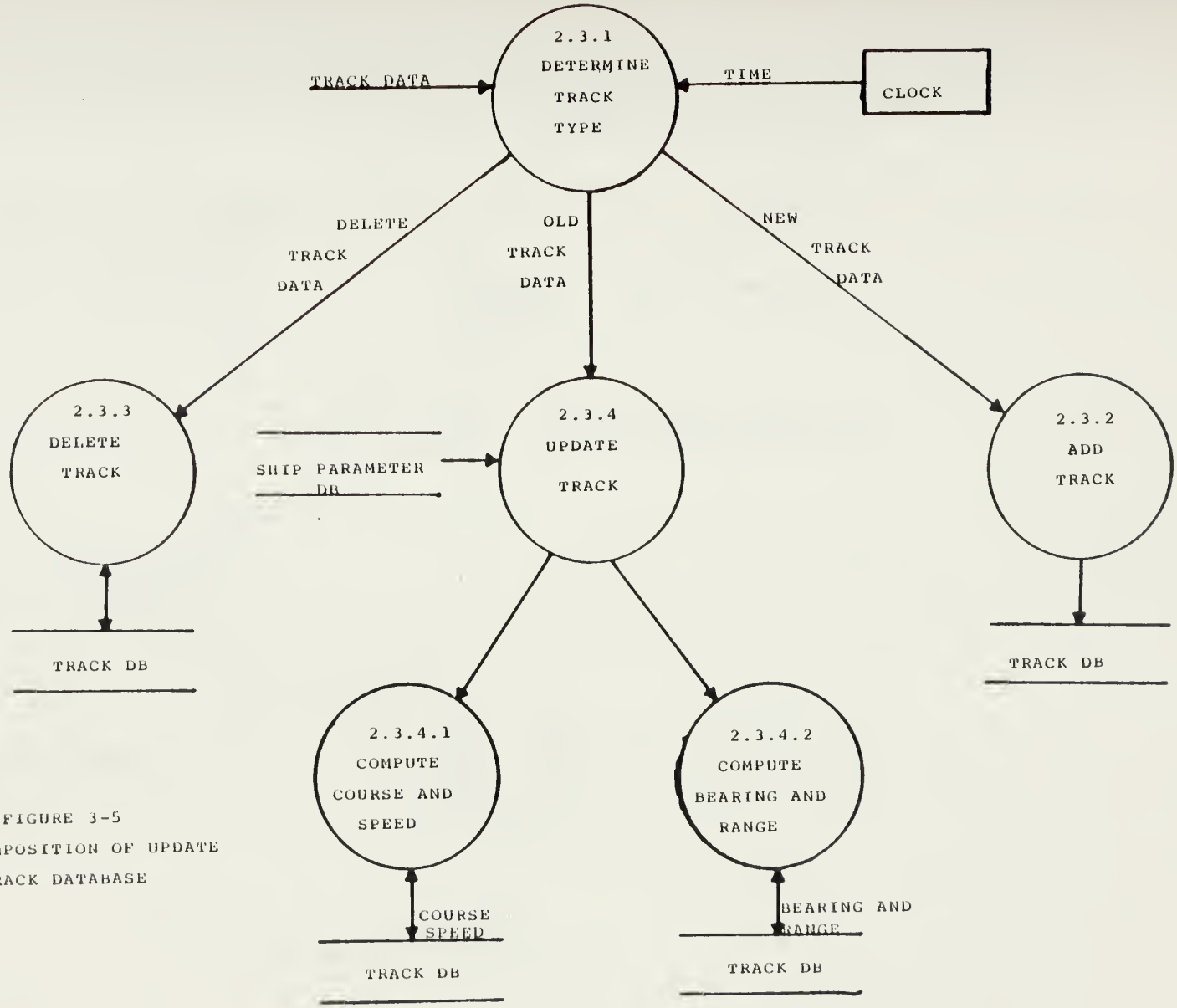


FIGURE 3-5
DECOMPOSITION OF UPDATE
TRACK DATABASE

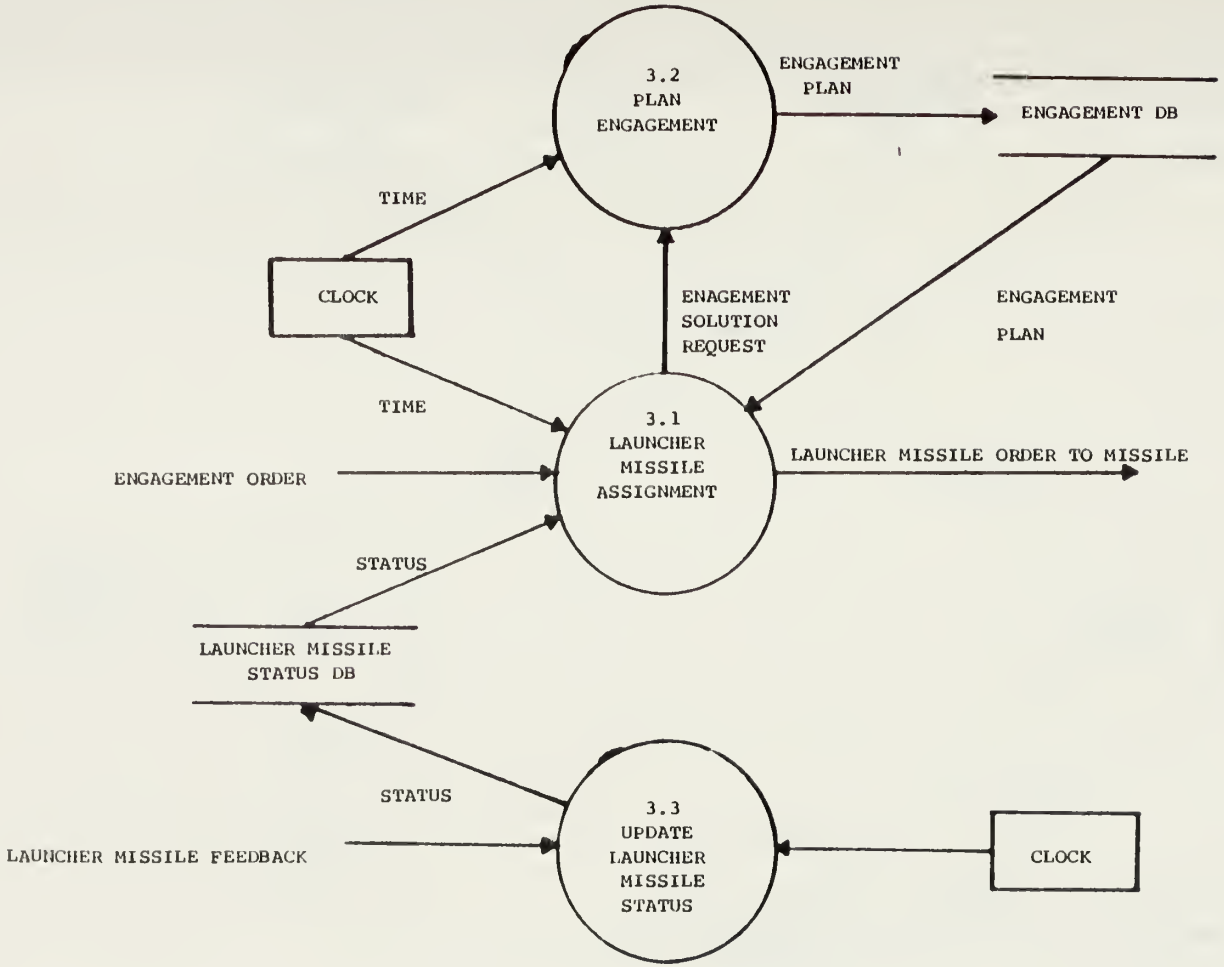


FIGURE 3-6

DECOMPOSITION OF ENGAGEMENT PLAN

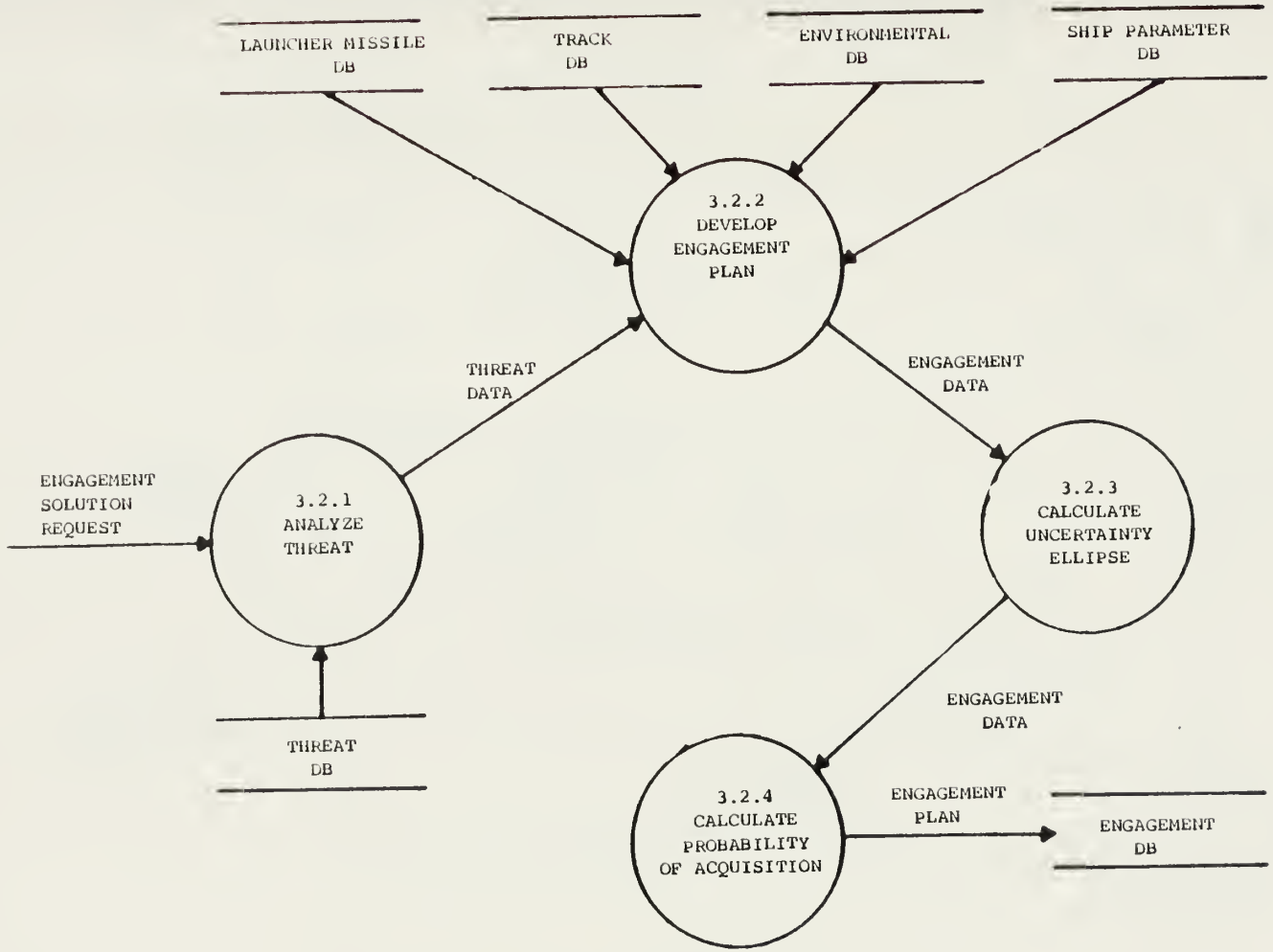


FIGURE 3-7
PLAN ENGAGEMENT DIAGRAM

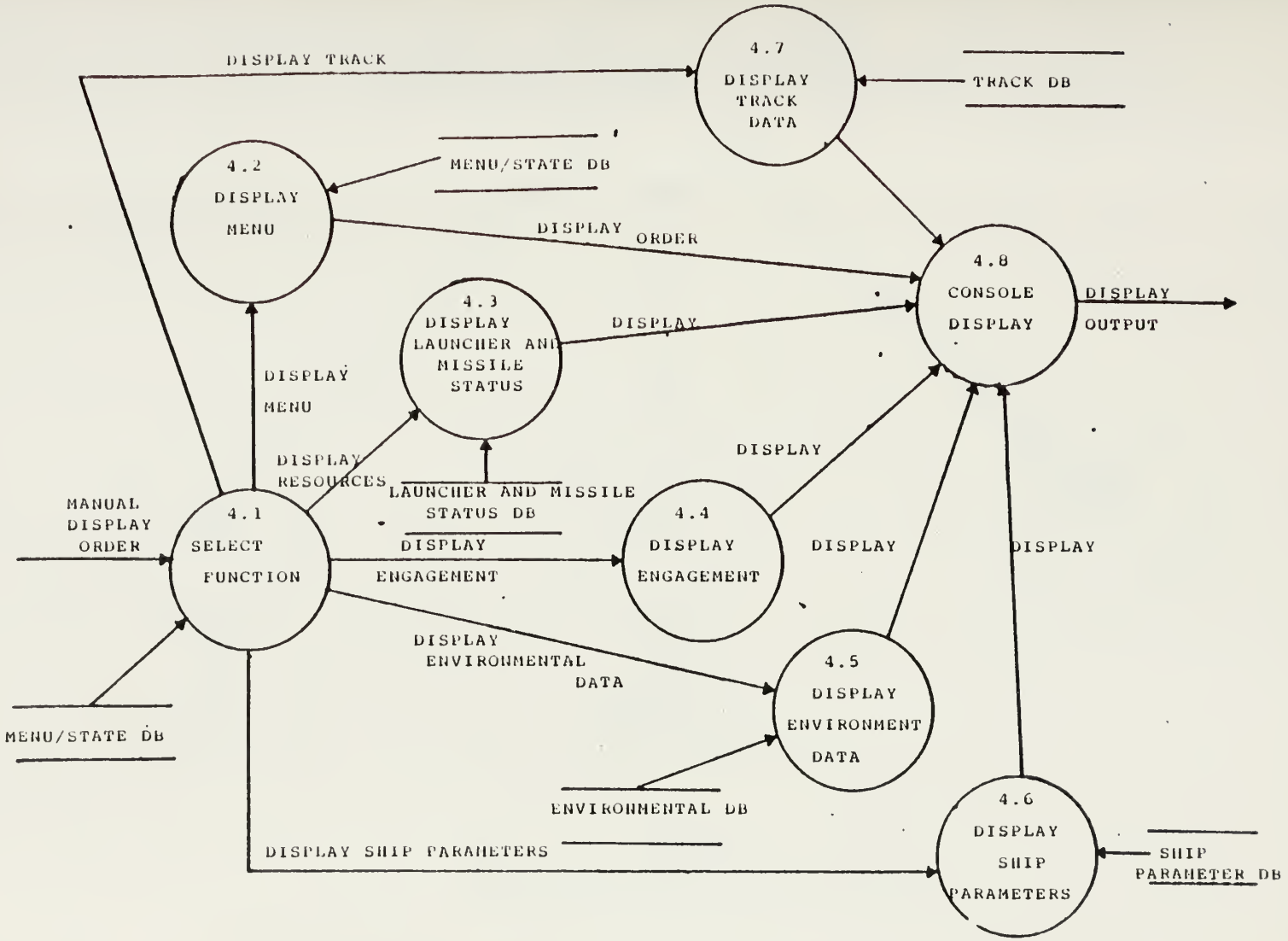


FIGURE 3-8.

DECOMPOSITION OF DECODE OUTPUT

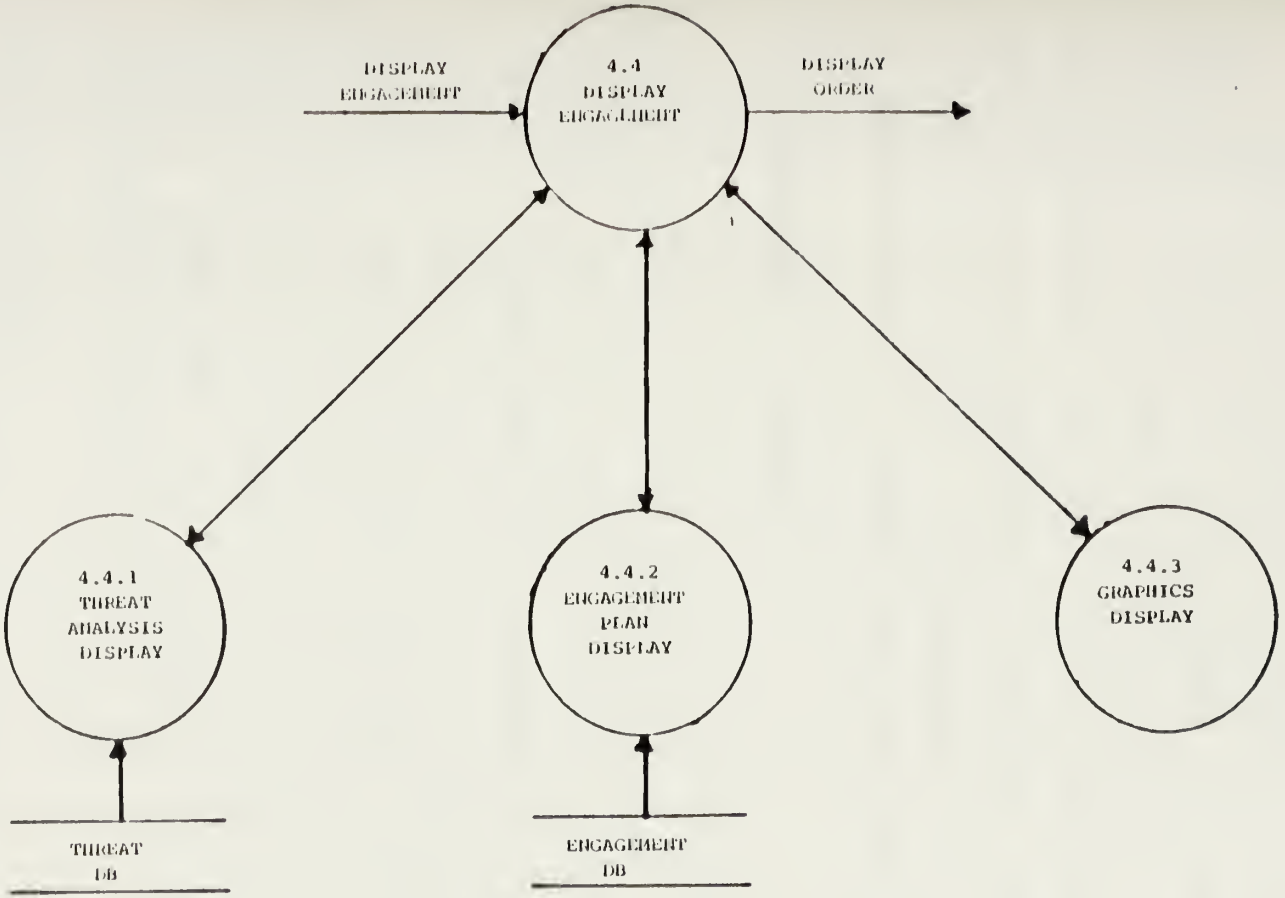


FIGURE 3-9

DECOMPOSITION OF DISPLAY ENGAGEMENT GRAPHICS

IV. TRANSFORMATION ANALYSIS OF DATA FLOW DIAGRAMS

The data flow diagrams presented in Chapter III define the flow of data within the system. The sources of data, the transaction that occurs and the destination of the data have been identified. The next step in the design process is to add a control mechanism to monitor the exchange and manipulation of data. To accomplish this task, a hierarchy of modules based on the DFD's must be established.

A. TRANSFORMATION ANALYSIS

The transformation analysis is achieved by using the DFD's as presented in the previous chapter. Each bubble in the DFD's initially becomes a module within a hierarchy of modules. Through heuristic refinement of these modules and the imposing of lines of control, a hierarchy of modules is achieved. The initial module hierarchy is presented in Figure 4-1. This is the simple transition from DFD's to modules. With control of the modules (e.g., what module "uses" what module) established, the refinement of the initial hierarchy is presented in Figure 4-2.

B. PROCESS INPUT

Any input to the system is processed by the Process Input module with its associated submodules. Ship parameter, environmental data, track data and threat data are all

processed in this module. Basically this group of modules serve only to update data bases from which other modules use this data. The method in which data is entered into the system is abstracted; it can either be done automatically or manually. Process Input is shown in Figure 4-3.

C. PLAN ENGAGEMENT

This group of modules is potentially the most complex within the system. When queried for an engagement plan, these modules must use all available information to either analyze a manual plan or develop an automated plan. Launcher missile status (i.e., what type of missiles are available in which launcher) is closely related to the type of plan developed so these modules appear on the same level of the hierarchy. Plan Engagement is shown in Figure 4-4.

The refinement from the initial DFD hierachy to the final hierarchical diagram is quite minor in nature. Bearing/range and position were included into one vice two modules. For launcher/missile assignment, this module was moved to the same level as plan engagement. This allows the system a distinct advantage over the previous design. The updated hierarchy now permits a manual launcher/missile assignment to be made without having the engagement analyzed. This enables a weapon to be fired quickly if the tactical situation so dictates. An automated engagement plan is also possible if so

desired based again on the tactical situation and/or time constraints.

D. DISPLAY

The display modules perform all the necessary functions to enable the WCIP operator to access all needed information. The menu display module simply presents the different options available to the operator. The remaining modules perform the tasks that their names imply. Launcher/missile status, environmental data, engagement data, ship parameter data and track data are all functions the operator may choose. Display is shown in Figure 4-5.

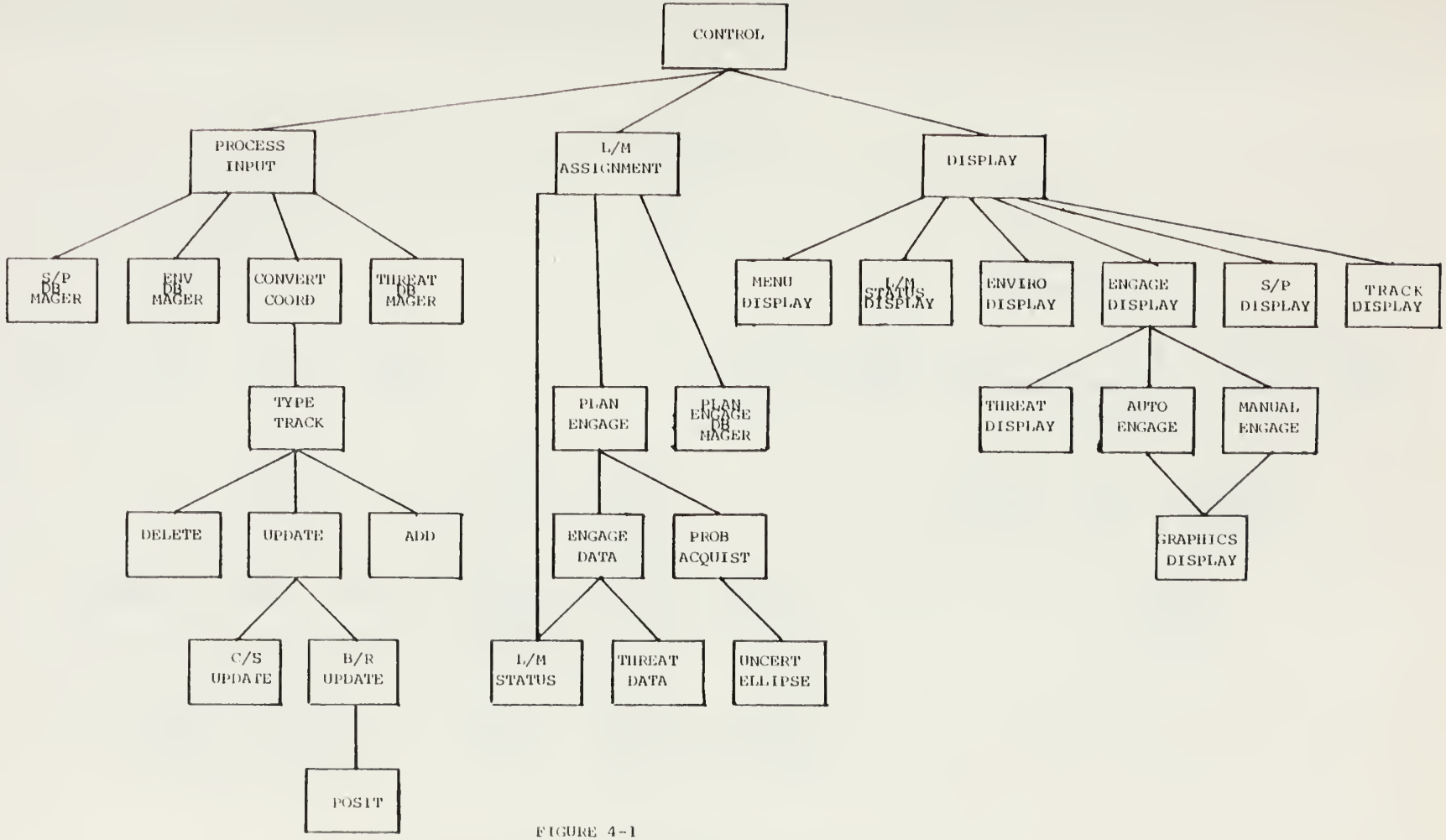


FIGURE 4-1

FIRST CUT TRANSFORM ANALYSIS

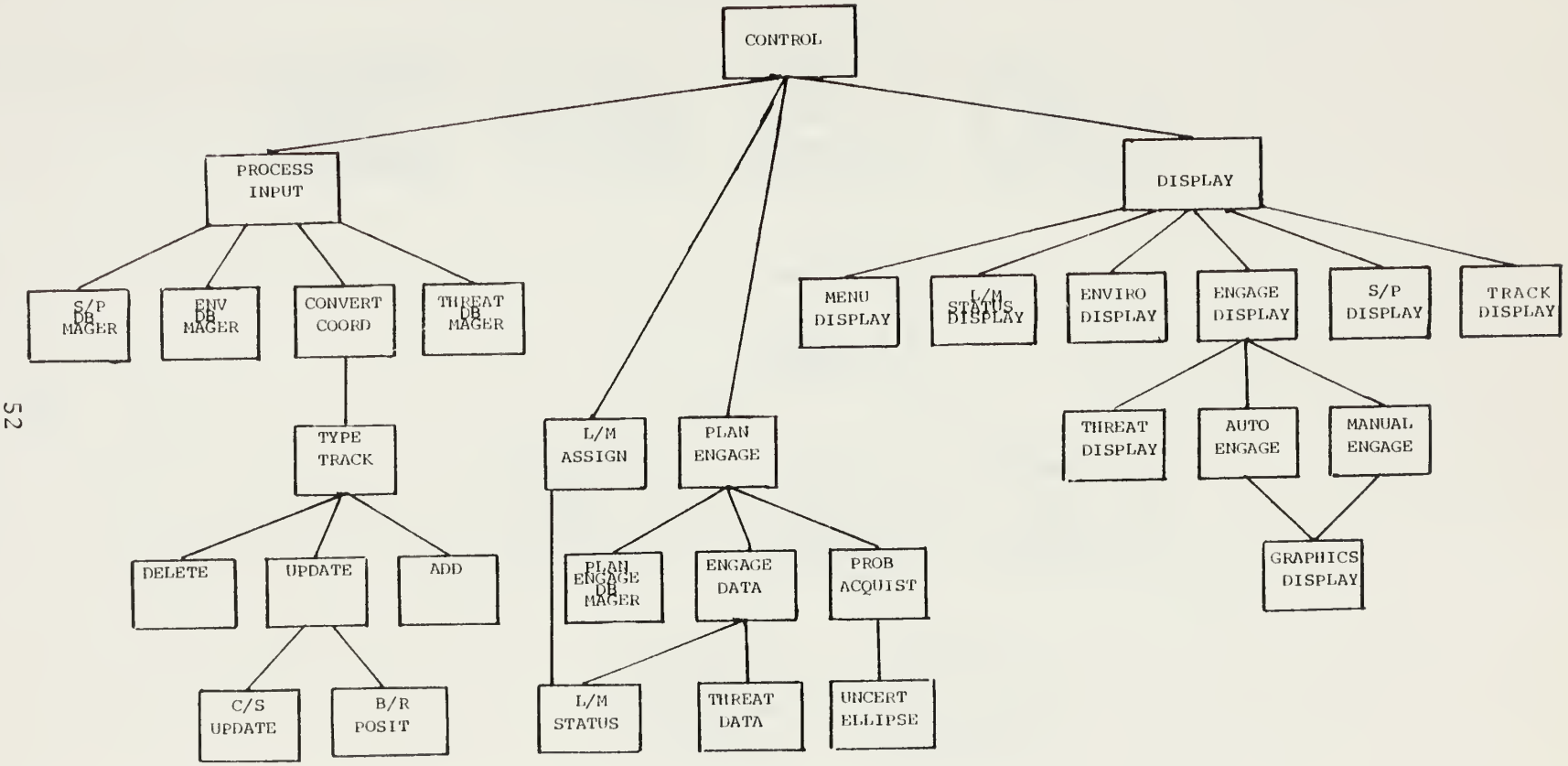


FIGURE 4-2

REFINEMENT OF TRANSFORM ANALYSIS

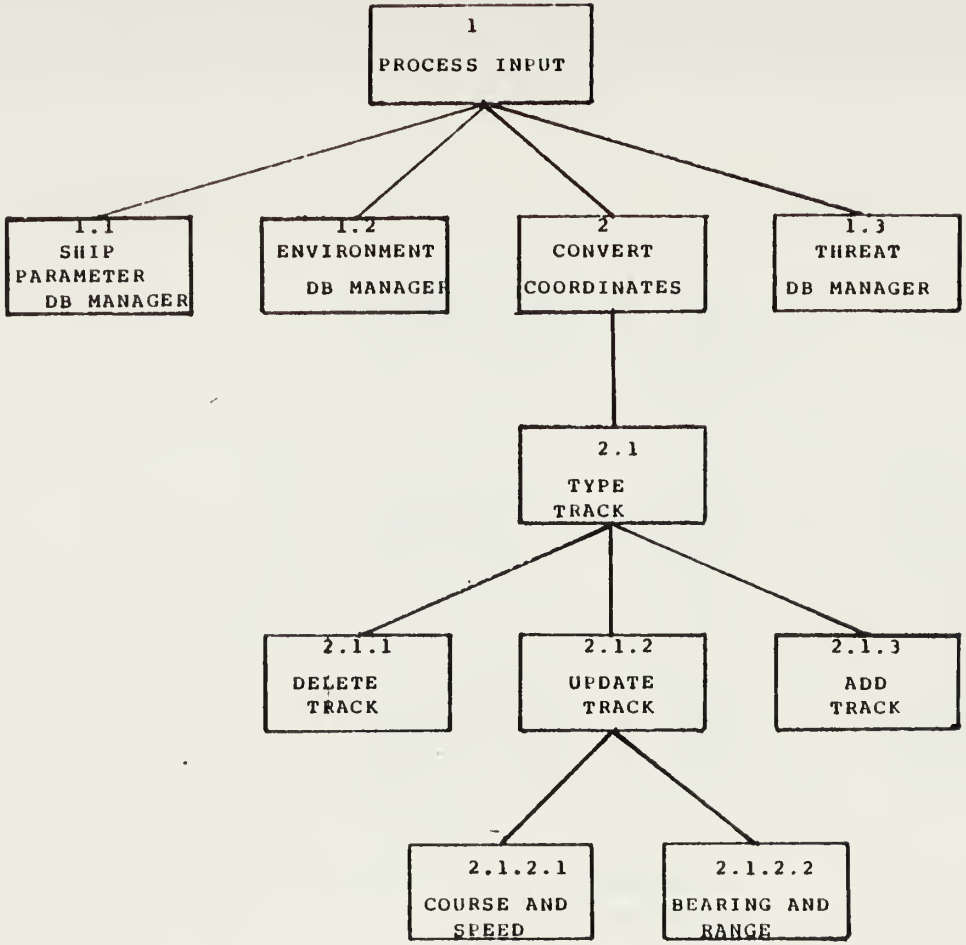


FIGURE 4-3

PROCESS INPUT

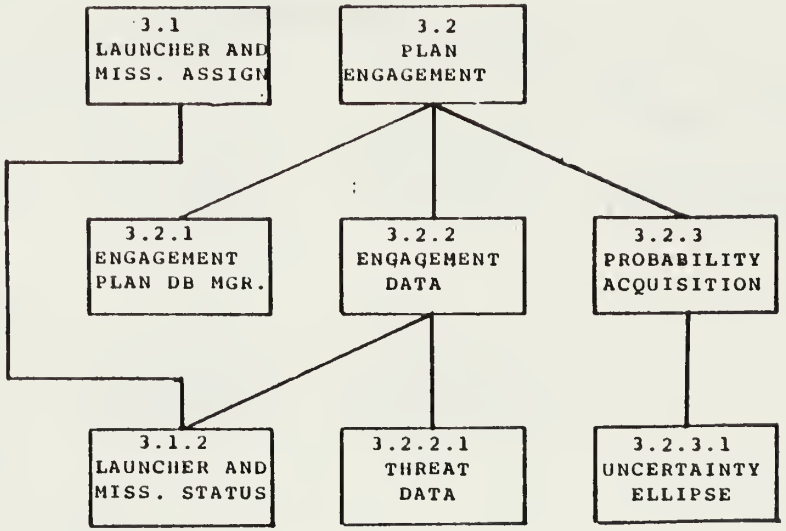


FIGURE 4-4

PROCESS ENGAGEMENT

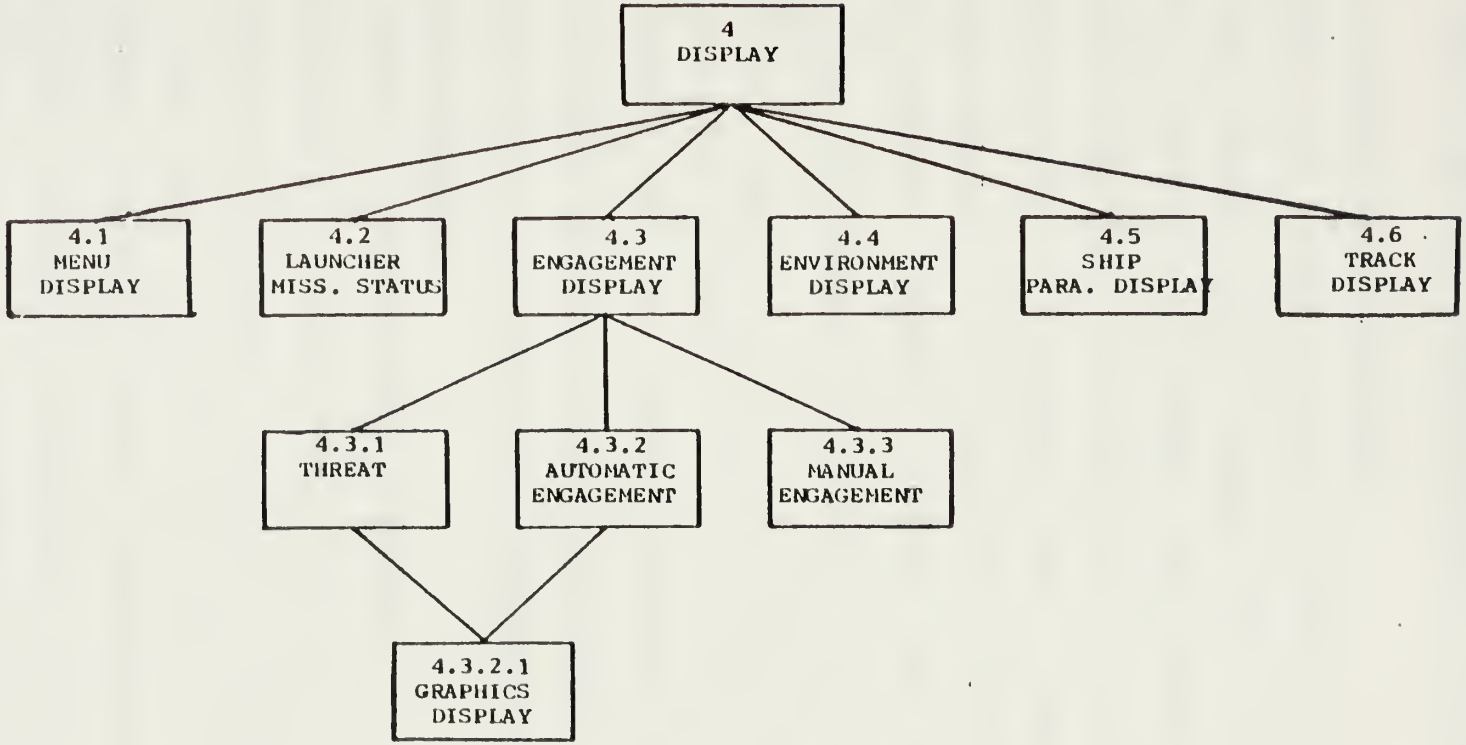


FIGURE 4-5

PROCESS DISPLAY

V. TRANSITION FROM TRANSFORM ANALYSIS TO SDL ADA

A. ADA AS A SYSTEM DESIGN LANGUAGE

Describing the product of the transform analysis of Chapter IV in a System Design Language (SDL) is the next step in the software development process. Ada was selected as the System Design Language to be used on this project. The reasons Ada was chosen are twofold. Ada is sponsored by the Department of Defense and is designed for the programming of all embedded systems. Secondly, Ada not only embodies many modern software development principles, but also enforces them. An SDL written in Ada does not require implementation in Ada; the implementation can be in any programming language. Further discussion of Ada as both a System Design Language and Program Design Language is treated in the thesis of LCDR George Wylie and LT Tom Watt, Utilization of Ada as a Program Design Language.

The purpose of a System Design Language is to show what program units need to be constructed and what interfaces each unit provides and requires. Ada as the System Design Language in this thesis is only the first iteration of a complex process. Some of the more detailed aspects of the design, such as a complete ennumeration of all elements in each record, are left for later design iterations. This preliminary design is abstracted to a higher level

emphasizing the design structure. A possible follow-on thesis could attempt further refinement of this design.

As shown in the transition from Data Flow Diagrams (Chapter III) to the Transform Analysis (Chapter IV), slight modifications to the design were necessary. Here again in the transition from Transform Analysis into an SDL using Ada, minor modifications have been made to the design. These modifications allow better grouping of Ada's packages.

B. AN OVERVIEW OF ADA

Before the Ada specifications of this program are discussed, some general Ada concepts will first be presented. This chapter will not attempt to give a detailed explanation of Ada but only a brief introduction to allow understanding of the Ada design shown in Appendix E. All Ada program units have a two-part structure: a specification and a body. The specification identifies the information a program unit can access and the interfaces to other units in the program. The body of the program unit contains the implementation details. The specification and body portions of a program unit may be written and compiled separately. At the SDL level used in this thesis, only the specifications for the program units will be written.

1. Subprograms

Ada has three forms of program units: subprograms, packages, and tasks. Subprograms are the basic unit of

execution in an Ada system and may be either procedures or functions. The design used here contains only procedures. Procedures are used to perform certain specific functions. The six procedures used in this SDL are for Control, Automatic Engagement, Manual Engagement, Graphics, Probability of Acquisition, and the Uncertainty Ellipse. Following the name of the procedure are the parameters that are used and their data type. (This terminology is the same for packages and tasks). The terms "in", "out", or "in out" specify the mode, or direction, of the data flow into the procedure.

2. Packages

Packages permit the encapsulation of a group of logically related entities. In this design, packages are used to encapsulate all input functions, all display functions, the manual engagement, the automatic engagement, and all of the data bases. Packages can be contained within other packages as is shown by the package Engagement Display inside the Display package, and by the data base packages inside the Data Base Manager package. By separating specific groups of related program units into packages, these packages can be reused by other programs. This way a library of packages can be established to be used for the development of other systems. Transportability of software and creation of a library of packages is one of the main goals of Ada. The

functions performed by the package Update and Display that manipulate data bases are common to many other embedded systems. This design, when finalized, can be exported to support those systems also.

3. Tasks

Tasks form the last class of Ada program units. Tasks can be viewed as independent and concurrent operations of program units. Tasks are similar to subprograms in that they perform some action and are not collections of entities like a package. Tasks communicate by means of an entry statement that is similar to a subprogram call statement. A task can have more than one entry statement as seen in the appendix by the Update-Track task. In this design, tasks are used to control the updating and accessing of all data bases described in the transform analysis of Chapter IV.

All inputs to data bases are channeled through the Update package. These inputs can come from manual or automated means. Although not available today, it is felt that the inputs to the Ship Parameter, Environment, and Launcher and Missile Status data bases can be automated in the future. This design using tasks will allow for these modifications by simply adding another entry statement for each task. Use of tasking in the design would also support a separate processor for the Update package. This processor could be devoted solely to the update of data bases. The

tasks within the package can be prioritized although this is not done in the appendix.

In the Display package, tasks are again used extensively. The display of information from each data base can be accessed through a task. A separate processor can again be assigned to perform only display functions; however, this is not required. If the design were implemented taking full advantage of tasking, three processors could operate concurrently--one to update data bases, one for display of data bases, and one to determine the engagement plans.

The Update and Display packages pass information to the data bases through the use of entry statements in tasks. The data bases are contained within the Data Base Manager package. Updates and accesses to the Data Base Manager package are allowed by entry statements that correspond to those used in the Update and Display packages. Although the terminology is confusing, it is a way of separating the data bases themselves from the functions that update and access them.

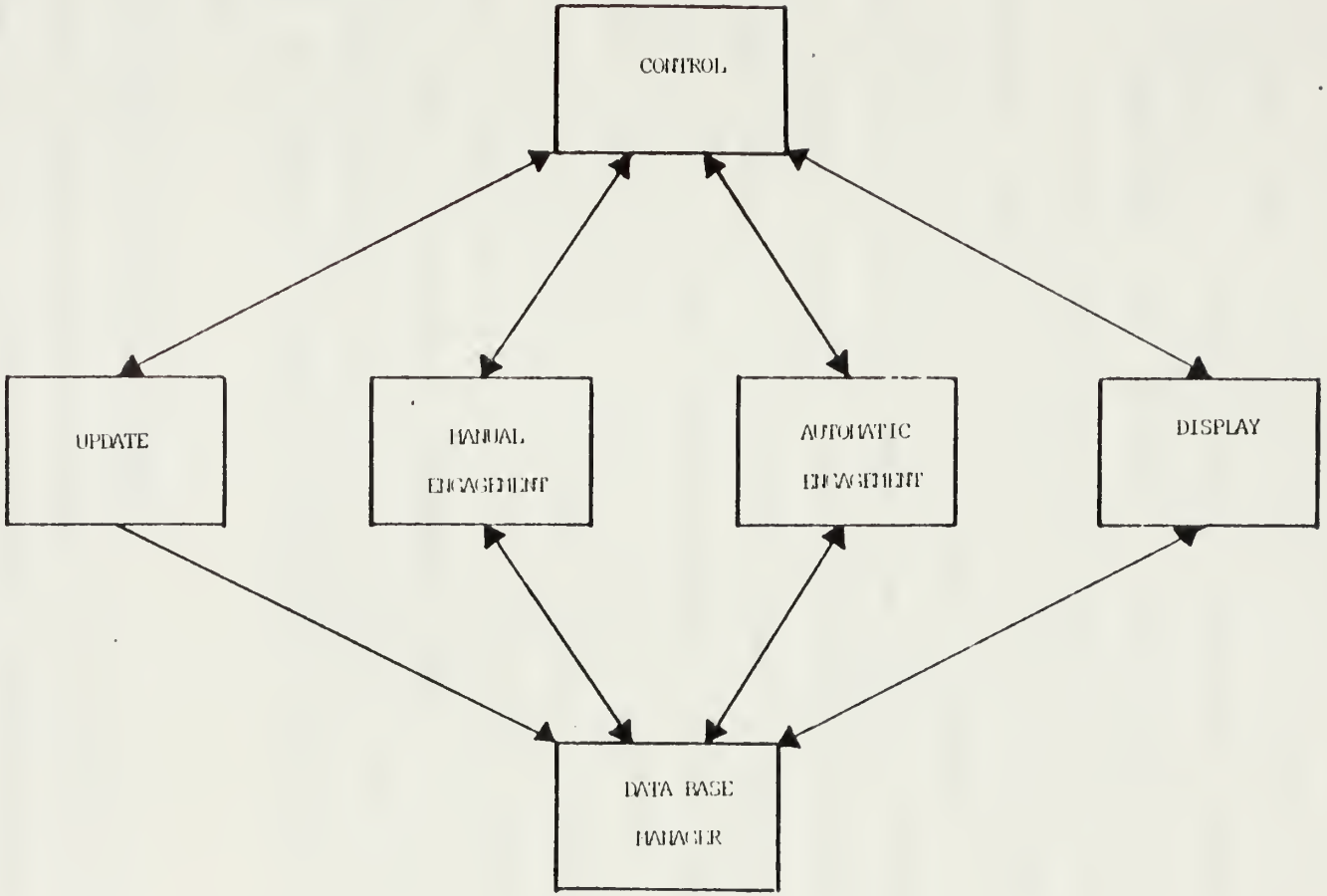
Use of tasks is a new concept to many designers. The principles of when to use tasks and in what designs have not been established. Tasks are suggested for this design to allow concurrent operation and the ability to update data bases by manual or automated means. Tasks used in the Update and Display packages have been grouped together and not

declared in separate packages as some textbooks recommend. It is felt that the functions of tasks in this design are similar enough to allow encapsulation of many within a single package. The best method for calling tasks and associated control is also not well defined in the literature. Further research in the use of tasks is required to solve these questions.

C. SYSTEM DESIGN LANGUAGE ADA

In the design using the System Design Language only two modifications were made to the structure of the transform analysis. First the Launcher Missile Status module was moved to the Update package. This is to allow groupings of all tasks that required update into one common package. The second modification was the movement of the Threat Display module up one level. Initially Threat Display was considered a subunit of Engage Display. In the grouping of tasks into packages, it was decided that all tasks would be grouped in the Display package and only procedures would be contained in the Engagement Display package. Both changes group related program units into packages.

The structure of the design remains the same as shown in the transform analysis of Chapter IV. A general diagram of the structure is shown in Figure 5-1. The only addition to the structure of the transform analysis is the addition of a Data Base Manager package on the bottom of the hierarchy. A



PROGRAM DESIGN STRUCTURE
FIGURE 5-1

control or main procedure is used to call subordinate packages. The specifications of the control procedure are not given in the appendix. An Update package is used to update all of the data bases identified in the transform analysis. Manual Engagement is identified as a separate package from Automatic Engagement even though its specification may seem trivial. It was felt that the functions of these two units are dissimilar enough to warrant separate packages. The Display package allows the operator to display the data contained in any of the data bases. The display of Engagement Plans is a separate package within the Display package.

The last package is the Data Base Manager package. It is here that the data bases and data types are defined. Each data base is defined as a record within a specific package. Some of these records will exist within a complex data structure if the data base has more than one element as in multiple tracks or multiple engagement plans. Also contained within each package are the entry statements that allow other packages to update or access information contained within a data base.

D. AUTOMATION OF THE SDL PROCESS

In this present design method, a gap exists between the development of the transform analysis and the System Design Language. This gap can be narrowed if the transition between these steps can be automated. Automation can begin with the

description of modules. An attempt was made here to define each module in a format that would readily convert to Ada. Module descriptions included the subheadings of "Objects used by the module" and "Operations performed on objects" that support Object Oriented Design using Ada [Ref. 1]. If these module descriptions can be entered into a computer that translates them into specifications for program units, this transition can be greatly assisted. The operator can enter the module description and request specifications for a subprogram, package or task. As the transform analysis will most likely not map directly into Ada program units, some modifications will have to be made.

Not only can the computer assist in the writing of specifications but all interfaces between units can be verified. The program can ensure that all units called do exist and have the correct data types. Additionally, the computer can detect any cases where terminology is not consistent. The output of this program can be specifications for each program unit with correct syntax. A program such as this could simplify the complex transition from transform analysis to System Design Language.

The major benefit of automating this process is the ability of the computer to check interfaces. Interfaces between program units are not complex on this design with thirty-one modules, but increases in complexity as the number

of modules rises. More detailed designs involving hundreds of modules can be assisted by a program that can verify interfaces between modules as correct. Although a program such as this is not available today, IBM, TRW, Harris and Norden are working on versions of ADA that will provide capabilities such as those mentioned [Ref. 5].

E. SUMMARY

The specifications shown in Appendix E are a first attempt to transition from a transform analysis into a System Design Language. The next step after further refinement of the specifications will be to write the code for the body of each package. When the specifications are complete, writing the code can become a relatively trivial matter. The major difficulties that arise in the design of a complex system should be resolved at the design level and not while coding. This philosophy is supported by Ada. All program specifications must be completed before program bodies can be written.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. DESIGN SUMMARY

In developing a design methodology for complex embedded systems, the authors have found that a combination of design techniques is the best approach.

In designing any system, the key to successful implementation is to keep the design simple. Through the use of DFD's the basic flow of information and data within the system can be analyzed. It is important to note that the DFD's must be constructed without any attached control structure. Controlling the flow of data too early in the design process complicates the design. Mistakes that occur early in the design (e.g., DFD's) are compounded throughout the design steps.

Through the use of a transform analysis, the initial DFD's are used with the added feature of a control structure. This enables a modular hierarchy to be formed. The user hierarchy (what module uses what module) presents a clear presentation of how the entire system interacts. With a modular hierarchy established, the modules may be formed into packages using Ada as the SDL.

B. TRANSITION TO SYSTEM DESIGN LANGUAGE

The transition from the Transform Analysis to a System Design Language (SDL) is the most difficult step in the design process. The difficulty lies in the lack of clear, well-defined standards for use of a SDL. Ada was chosen as the SDL for this design for several reasons: Ada is sponsored by the Department of Defense; it was created to be used as a design language; and it embodies many software engineering principles. The Ada design used in this thesis is abstracted to a very high level and only the specification portion of the code is presented (Ada program units consist of two parts, the specification and body of each unit).

In the transition from the structure of the Transform Analysis to the SDL level, an additional level was added to the bottom of the hierarchy. This bottom level contains the data bases that are used by the higher level packages. Although the design might seem simpler at first if the data bases were enclosed in the packages that require their use, encapsulation in separate packages does have advantages. If changes are required to the data bases at a later date, these changes can be isolated to the single package that contains the data bases and not affect the packages that use the data base. This also allows identifying the Update and Display packages as separate generic units that can be used in other programs.

Tasks are used extensively in the specifications of this program for two reasons. First, because they would allow separate functions such as updates, display, or engagement plan generation to be performed by separate processors. Secondly because it allows updating by more than one means (such as manual and automatic) by simply adding additional entry statements.

This transition to a SDL can be eased if the process can be automated. A program is needed that will take module descriptions generated at the Transform Analysis level along with a general program structure and generate the SDL specifications. This program can also ensure correct module interface definitions. Part of the difficulty in automating this process is defining what items are needed for module definitions. The module definitions used in this thesis were designed with the thought of automating this process in mind.

C. RECOMMENDED FOLLOW-ON WORK

The authors recommend research be conducted in the following areas to support the HSCLCS improvement and general design methodology:

- Discuss the design aspects of Harpoon that are directly transferable to the Tomahawk cruise missile and other cruise missile follow-ons.
- Develop an automated process to verify interfaces between modules. The program could be used to generate an SDL from module descriptions and a general design structure.

- Explore and develop the algorithm or model associated with the uncertainty ellipse.
- Define the parameters and algorithm necessary to develop the probability of acquisition module.
- Develop the automated engagement plan analysis algorithm.
- Further refine the SDL Ada and construct the bodies of each package shown in Appendix E.

APPENDIX A

GLOSSARY

Abstraction - A psychological notion that affords a view of a problem at some level of generalization without regard to irrelevant low level details. Use of abstraction allows the use of concepts and terms that are familiar in the problem environment without having to transform them to an unfamiliar environment [Ref. 6].

Abstract Interface - Allows inputs into or outputs from a module to match changes in inputs or outputs to only affect the abstract interface code, and not the code on the output side of the module. Trys to solve the embedded computer problem and keep the cost down.

Bubble Diagram - see Data Flow Diagram (DFD).

Data Flow Diagram (DFD) (sometimes called a bubble diagram) - A graphical tool used to depict data (information) flow. The DFD uses the following graphical symbols: labeled arrows to represent information flow, labeled circles called "bubbles" that represent processes (transformations), labeled boxes that represent information sources and sinks, and two labeled parallel lines that represent stored information [Ref. 6].

Data Base- A file of interrelated data that are stored together to serve one or more applications and that are independent of programs using the data [Ref. 1].

Data Structure - Dictates the organization, methods of access, degree of associativity, and processing alternatives for information [Ref. 6].

Embedded System Program - A computer program that is part of some larger entity and essential to the operations of that system. For example, the timer on a washing machine or the guidance system in a missile may have computer programs. These programs are considered to be embedded.

Function - Name given to one or more statements that perform a specific task. Results in a value being assigned to its name upon execution of that function [Ref. 6].

Information Hiding - Specification and design of modules so that information (procedure and data) contained within a module are inaccessible to other modules that have no need to know the information [Ref. 6].

Interface - Communications between modules governed by a set of assumptions one module makes about another [Ref. 4].

Module - A separately addressable element of a program [Ref. 6].

Modular Design - A logical partitioning of software into elements that perform specific functions or subfunctions [Ref. 6].

NTDS - Naval Tactical Data System. Allows a data link between various platforms. Real time information is passed via the link.

Package - A program unit specifying a collection of related entities such as constants, variables, types, and subprograms. The visible part of the package contains the entities that may be used from outside the package. The private part of the package contains structural details that are irrelevant to the user of the package but that complete the specification of the visible entities. The package body contains the implementation of the subprograms or task (possible other packages) specified in the visible part [Ref. 1].

Packaging - Alludes to the techniques used to assemble software for a specific processing environment or to ship software to a remote location [Ref. 6].

Probability of Acquisition - Calculated probability of seek-head acquisition of intended target based upon information available.

Requirements Analysis - Third step in the software engineering procedure, last of the planning phase steps. Provides a foundation for the development of the software by uncovering the flow and structure of information. Describes the software by identifying interface details, providing an in-depth description of functions; determining design con-

straints and defining software validation requirements. Establishes and maintains communication with the user and the requester so that the above two objectives may be satisfied [Ref. 6].

Software Engineering - Software implementation of a problem solution approached by using a set of techniques that are application independent. These techniques are (1) a well-defined methodology that addresses a software lifecycle of planning, development, and maintenance, (2) an established set of software components that documents each step in the life cycle and shows traceability from step to step, and (3) a set of predictable milestones that can be reviewed at regular intervals throughout the software lifecycle [Ref. 6].

Software Requirements Specification - The deliverable of the software requirements analysis phase of the software engineering process. Contains introduction, information description, functional description, validation criteria, bibliography, and appendix [Ref. 6].

Software Plan - Second step in the software engineering process. Provides a framework that enables the manager to make reasonable estimates of resources, cost, and schedule.

Stepwise Refinement - The architecture of the program is developed by successively refining levels of procedural detail. Early software top-down design procedure proposed by Niklaus Wirth [Ref. 6].

Subprogram - An executable program unit, possibly with parameters for communication with its point of call. A subprogram declaration specifies the name of the subprogram and its parameters; a subprogram body specifies its execution. A subprogram may be a procedure, which performs an action, or a function, which returns a result [Ref. 1].

Subordinate Module - A module controlled by another module [Ref. 6].

Superordinate Module - A module that controls another module [Ref. 6].

System - A collection of elements related in a way that allows accomplishment of some tangible objective [Ref. 6].

System Analysis - Comprised of a number of tasks that define what must be accomplished, whether accomplishment is feasible, and what the cost-benefit of accomplishment will be [Ref. 6].

System Specification - First deliverable in the computer system engineering process. Contains introduction, functional description, allocation, constraints, cost, schedule of system development known at the time of the completion of the system specification [Ref. 6].

Task - A program unit that may operate in parallel with other program units. A task specification establishes the name of the task and the names and parameters of its entries; a task body defines its execution. A task type is a

specification that permits the subsequent declaration of any number of similar tasks. A task is said to depend upon the unit in which it is declared (subprogram body, task body, or a library package body). A unit is not left until all dependent tasks are terminated. A task is completed if it is waiting at the end of its body for any dependent tasks or is aborted but not yet terminated. A completed task cannot be called. A terminated task is, in a sense, the same as a dead task (it is no longer active) [Ref. 1].

Transform Flow - Flow that can be characterized by an afferent flow (i.e., incoming data), transformation (i.e., some change or action on the data, and efferent flow (i.e., output flow) with no regard to the number of flow paths [Ref. 6].

Uncertainty Ellipse - A probability associated with a track (or target) that its position is within a given geographical location.

APPENDIX B

DATA BASE DESCRIPTIONS

This appendix contains the data base descriptions used in the composition of the Data Flow Diagrams. The seven (7) data bases are:

1. Engagement Data Base
2. Environmental Data Base
3. Launcher and Missile Status Data Base
4. Menu/State Data Base
5. Ship Parameter Data Base
6. Threat Data Base
7. Track Data Base

1. Data Base Name: ENGAGEMENT DATA BASE

2. Purpose: This data base will contain a track name and associated engagement plans for that track. The engagement plan may be generated automatically by the computer or manually.

3. Data Base Users:

   a. Write Access: Calculate Probability of Acquisition

   b. Read Access: Engagement Plan Display

4. Data Base Elements:
   Track name
   Type engagement plan (manual or automatic)
   Number of missiles to fire
   Sequence of firing missile(s)
   Type of missile to use
   Flight path
   Waypoints

5. Operations on Data Base: Update
   Display

1. Data Base Name: MENU/STATE DATA BASE

2. Purpose: This data base will contain the menus for program operation and also provide the states allowable from any operation. That is it will provide the menus necessary to access all aspects of the program.

3. Data Base Users:

   a. Write Access: None

   b. Read Access: Display module

4. Data Base Elements: Undetermined at this time

5. Operations on Data Base: Operator can select desired menu item

1. Data Base Name: SHIP PARAMETER DATA BASE

2. Purpose: This data base will maintain all pertinent information pertaining to one's own ship. The design allows for this information to be input manually or automatically.

3. Data Base Users:

   a. Write Access: Ship Parameter Data Base Manager

   b. Read Access: Update track
      Plan Engagement
      Ship Parameter Display

4. Data Base Elements: Course
   Speed
   Position

5. Operations on Data Base: Update
   Display

1. Data Base Name: THREAT DATA BASE

2. Purpose: This data base is to contain a list of hostile surface vessels by name and class. Associated with each class of vessel will be the weapons platform, ECM capabilities, and optimum engagement plan for attacking that vessel. (The security of this information must be considered when designing the Threat Display and Threat Data Base Manager modules).

3. Data Base Users:

   a. Write Access: Threat Data Base Manager

   b. Read Access: Threat Display
      Analyze Threat Data

4. Data Base Elements: Ship name
   Ship class
   Weapons
   ECM equipment
   Engagement plan recommended

5. Operations on Data Base: Update
   Display

1. Data Base Name: LAUNCHER AND MISSILE STATUS DATA BASE

2. Purpose: This data base will keep track of the number and type of missiles available for launch. The data base will be updated by feedback from the launcher.

3. Data Base Users:

   a. Write Access: Launcher Missile Status

   b. Read Access: Launcher Missile Assignment
      Plan Engagement
      Launcher Missile Status Display

4. Data Base Elements: Launcher number
   Missile type

5. Operations on Data Base: Update
   Display

1. Data Base Name: ENVIRONMENTAL DATA BASE

2. Purpose: Contains the current state of weather; visibility, sea state, winds, etc.

3. Data Base Users:

   a. Write Access: Environment Data Base Manager

   b. Read Access: Environment Display
      Plan Engagement

4. Data Base Elements: (This list is not meant to be all inclusive but merely a sample of information that would be beneficial)
   Visibility
   Sea state
   Winds-direction and speed
   Humidity
   Temperature
   Cloud coverage

5. Operations on Data Base: Update manually or automatically
   Display

1. Data Base Name: TRACK DATA BASE

2. Purpose: This data base will contain the position of all tracks and pertinent information pertaining to that track.

3. Data Base Users:

   a. Write Access:
      - Delete Track
      - Add Track
      - Course and Speed Update
      - Bearing, Range, and Position Update

   b. Read Access:
      - Display Track Data
      - Plan Engagement

4. Data Base Elements:
   - Type track (friend or foe)
   - Class of vessel
   - Bearing
   - Range
   - Position (Lattitude and Longitude)
   - Course
   - Speed

5. Operations on Data Base:
   - Update
     - add
     - delete
     - change bearing range or position
   - Display

APPENDIX C

MODULE DESCRIPTIONS

This appendix contains the module descriptions of the modules shown in Figures 4-1 through 4-5. The thirty-one (31) module descriptions are:

1 - 0	Control
2 - 1	Process Input
3 - 1.1	Ship Parameter Data Base Manager
4 - 1.2	Environmental Data Base Manager
5 - 1.3	Threat Data Base Manager
6 - 2	Convert Coordinates
7 - 2.1	Type Track
8 - 2.1.1	Delete Track
9 - 2.1.2	Update Track
10 - 2.1.2.1	Course and Speed Update
11 - 2.1.2.2	Bearing, Range, and Position Update
12 - 2.1.3	Add Track
13 - 3.1	Launcher and Missile Assignment
14 - 3.1.2	Launcher and Missile Status
15 - 3.2	Plan Engagement
16 - 3.2.1	Plan Engagement Data Base Manager
17 - 3.2.2	Engagement Data
18 - 3.2.2.1	Threat Data
19 - 3.2.3	Probability of Acquisition
20 - 3.2.3.1	Uncertainty Ellipse
21 - 4	Display
22 - 4.1	Menu Display
23 - 4.2	Launcher and Missile Status Display
24 - 4.3	Environmental Display
25 - 4.4	Engagement Display
26 - 4.4.1	Threat Display
27 - 4.4.2	Automatic Engagement Display
28 - 4.4.2.1	Graphics Display
29 - 4.4.3	Manual Engagement Display
30 - 4.5	Ship Parameter Display
31 - 4.6	Track Display

1. Module Name: CONTROL, NUMBER 0

2. Module Purpose: The Control module calls all other modules and determines the program flow.

3. Subordinate Modules:
   Process Input (1)
   Launcher Missile Assignment (3.1)
   Plan Engagement (3.2
   Display (4)

4. Objects Used by the Module: Manual inputs

5. Operations Module Performs: Selection of subordinate modules to perform program operation.

1. Module Name: PROCESS INPUT, NUMBER 1

2. Module Purpose: Selects subordinate module to update corresponding data bases.

3. Subordinate Modules:
   Ship Parameter Data Base Manager (1.1)
   Environmental Data Base Manager (1.2)
   Convert Coordinates (2)
   Threat Data Base Manager (1.3)

4. Objects Used by the Module: Manual Inputs to update modules

5. Operations Module Performs: Selects appropriate subordinate module to update corresponding data base.

1. Module Name: SHIP PARAMETER DATA BASE MANAGER, NUMBER 1.1

2. Module Purpose: Update the Ship Parameter Data Base by either manual or automated means.

3. Subordinate Modules: None

4. Objects Used by the Module: Ship parameter input data

5. Operations Module Performs: Update of the Ship Parameter Data Base.

1. Module Name: ENVIRONMENTAL DATA BASE MANAGER, NUMBER 1.2

2. Module Purpose: Update the Environmental Data Base by either manual or automated means.

3. Subordinate Modules: None

4. Objects used by the Module: Environmental input or update data.

5. Operations Module Performs: Update of the Environmental Data Base.

1. Module Name: THREAT DATA BASE MANAGER, NUMBER 1.3

2. Module Purpose: Update the Threat Data Base by either manual means or through the use of a standard chip that can be periodically updated and sent to all ships with Harpoon capability.

3. Subordinate Modules: None

4. Objects Used by the Module: Data used to update the Threat Data Base.

5. Operations Module Performs: Update of the Threat Data Base.

1. Module Name: CONVERT COORDINATES, NUMBER 2

2. Module Purpose: To convert all inputs to update track data to common coordinates. The inputs can be manual, from own ship's tracking equipment, or from a NTDS link from other ships or platforms.

3. Subordinate Modules: Type Track (2.1)

4. Objects Used by the Module: Information used to update the Track Data Base, bearing, range, and position

5. Operations Module Performs: All sources of input for the Track Data Base are converted to common coordinates whether they are manual, NTDS, or from another source.

1. Module Name: NUMBER TRACK, NUMBER 2.1

2. Module Purpose: Type Track determines if the track is to be deleted from the data base, added to the data base, or some parameters of a present track are to be altered. These actions are performed by selecting the appropriate subordinate module.

3. Subordinate Modules: Delete Track (2.1.1)
Update Track (2.1.2)
Add Track (2.1.3)

4. Objects Used by the Module: Classification of type update to be performed, addition, deletion, or alteration to the Track Data Base.

5. Operations Module Performs: Selection of delete, update, or add track subordinate modules based on track type.

1. Module Name: DELETE TRACK, NUMBER 2.1.1

2. Module Purpose: To eliminate tracks from the data base that the operator determines are no longer useful.

3. Subordinate Modules: None

4. Objects Used by the Module: None

5. Operations Module Performs: Track identified to the Delete Track module is erased from the Track Data Base.

1. Module Name: UPDATE TRACK, NUMBER 2.1.2

2. Module Purpose: To update the information contained on tracks in the Track Data Base.

3. Subordinate Modules: Course and Speed (2.1.2.1)
Bearing and Range (2.1.2.2)

4. Objects Used by the Module: Bearing and range from a fixed point.
Ellapsed time.
Course and speed of own ship
Course and speed of target ship.

5. Operations Module Performs: Determines which subordinate module is to be called to perform the desired update.

1. Module Name: COURSE AND SPEED UPDATE, NUMBER 2.1.2.1

2. Module Purpose: To update the course and speed information on each track contained in the Track Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module:
   Course and speed of own ship.
   Ellapsed time.
   Bearing and range from a fixed point.
   NTDS link information.

5. Operations Module Performs: Determines course and speed of tracks based on bearing/range and ellapsed time when entered manually and updates Track Data Base. With automated track information available (e.g., NTDS) module updates Track Data Base with the given course and speed.

1. Module Name: BEARING, RANGE, AND POSITION UPDATE, NUMBER 2.1.2.2

2. Module Purpose: To update the bearing/range and position (Lattitude and Longitude) information on each track contained in the Track Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module:
   Own ship sensor information.
   NTDS link information.

5. Operations Module Performs: Determines bearing/range and position of tracks based on own ship sensor information and own ship position, when entered manually and updates Track Data Basae. With automated track information available (e.g., NTDS) module updates Track Data Base with the given bearing/range and position.

1. Module Name: ADD TRACK, NUMBER 2.1.3

2. Module Purpose: To allow new tracks to be input into the Track Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module: Course, speed, bearing, range, position, identification of track (friend or foe and ship class).

5. Operations Module Performs: Permits the addition of new tracks into the Track Data Base.

1. Module Name: LAUNCHER MISSILE ASSIGNMENT, NUMBER 3.1

2. Module Purpose: Allow the operator to bypass the engagement planning capabilities of the computer system and simply select and launch the desired missiles.

3. Subordinate Modules: Launcher and Missile Status

4. Objects Used by the Module: Inputs from operator identifying which launcher and missiles to be fired in which bearing, range and waypoints if desired.

5. Operations Module Performs: The Launcher Missile Assignment module allows the operator to manually select a launcher and missile to be fired in a given direction similar to the present capabilities of the Harpoon Weapons System. The automated engagement planning functions of this program are bypassed.

1. Module Name: LAUNCHER AND MISSILE STATUS, NUMBER 3.1.2

2. Module Purpose: To provide current information on what launchers (port and starboard) are ready to fire and which and what type missiles are ready for firing.

3. Subordinate Modules: None

4. Objects Used by the Module: Which launchers (port and starboard) are ready to fire Which and what type missiles are ready for fire

5. Operations Module Performs: The Launcher and Missile Status module receives automated inputs from each launcher on the status and type of all missiles. This information is used to update the Launcher and Missile Status Data Base. When querried by either the Launcher and Missile Assignment module or the Engagement Data module, the Launcher and Missile Status module can return the status of launchers and missiles.

1. Module Name: PLAN ENGAGEMENT, NUMBER 3.2

2. Module Purpose: To determine the optimum engagement plan for a given target.

3. Subordinate Modules: Plan Engagement Data Base Manager (3.2.1)
Engagement Data (3.2.2)
Probability of Acquisition (3.2.3)

4. Objects Used by the Module: Which track to plan the engagement for.

5. Operations Module Performs: The Plan Engagement module is the heart of this software program. This module determines an optimum engagement plan for desired targets. The targets that have an engagement plan computed can be identified either by a selective manual input or can be automatically generated for all contacts that are classified hostile (This latter process would require a slight modification to the design; an input to the Plan Engagement must be received from the type track module). Through access to the Threat Data Base, Launcher Missile Status Data Base the optimum plan for engaging a selected target can be computed. The functions performed by this module can greatly assist the Tactical Action Officer in the performance of his duties.

1. Module Name: PLAN ENGAGEEMENT DATA BASE MANAGER, NUMBER 3.2.1

2. Module Purpose: To update the Engagement Plan Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module: Engagement plan as generated by the Engagement Plan module.

5. Operations Module Performs: The Plan Engagement Data Base Manager enters all engagement plans that the Plan Engagement module generates into a Plan Engagement Data Base. This way a list of the engagement plans for all applicable targets is kept current in a data base.

1. Module Name: ENGAGEMENT DATA, NUMBER 3.2.2

2. module Purpose: The Engagement Data module supplies the data needed by the Plan Engagement module to generate the Engagement Plan.

3. Subordinate Modules: Launcher and Missile Status (3.1.2)
Threat Data (3.2.2.1)

4. Objects Used by the Module: Which launcher and missiles are ready to fire.
All pertinent information on hostile ship class, weapons, ECM equipment and best strategy for attack contained in the Threat Data Base.

5. Operations Module Performs: The Engagement Data module coordinates the passing of all data base information needed to generate an engagement plan to the Plan Engagement module. In this design that information is contained in the Launcher and Missile Status module and the Threat Data module.

1. Module Name: THREAT DATA, NUMBER 3.2.2.1

2. Module Purpose: To provide the information contained in the Threat Data module to the Engagement Data module when requested.

3. Subordinate Modules: None

4. Objects Used by the Module: All elements contained in the Threat Data Base, ship class, weapons, platform, ECM capability and best plan for attack.

5. Operations Module Performs: The Threat Data module provides the Engagement Plan module with all of the information that is contained in the Threat Data Base. This information is then used to determine the optimum engagement plan.

1. Module Name: PROBABILITY OF ACQUISITION, NUMBER 3.2.3

2. Module Purpose: To determine what the probability is that if a missile is launched at a given target that the missile can acquire and hit that target.

3. Subordinate Modules: Uncertainty Ellipse (3.2.3.1)

4. Objects Used by the Module: The figure generated by the Uncertainty Ellipse module.
Type missile to be launched and search pattern.
Type target to be attacked and its physical characteristics and ECM capabilities.

5. Operations Module Performs: The Probability of Acquisition module uses the type missile fired, range to target, and target characteristics to generate the probability that the missile can acquire and hit the given target. This information along with the value obtained from the Uncertainty Ellipse module is then passed to the Plan Engagement module where they become elements of the engagement plan maintained in the Engagement Plan Data Base.

1. Module Name: UNCERTAINTY ELLIPSE, NUMBER 3.2.3.1

2. Module Purpose: To determine the probability that a given missile, or set of missiles, fired at a specific target will sink that target.

3. Subordinate Modules: None

4. Objects Used by the Module:
   Number of missiles to be fired.
   Probability of acquisition of the target.
   Lethal capability of missiles fired.

5. Operations Module Performs: The Uncertainty Ellipse module takes the number and capabilities of missiles fired and combines these values with the probability of acquisition to generate the probability of a target kill. The Uncertainty Ellipse module can generate ellipses with assigned probabilities stating that total destruction of the target will occur if it is within one ellipse, 50% disability of the target will occur if it is within a second ellipse, etc. These ellipses will account for the fact that hostile target positions may not be completely accurate.

1. Module Name: DISPLAY, NUMBER 4

2. Module Purpose: To call subordinate modules as necessary to generate required displays.

3. Subordinate Modules:
   Menu Display (4.1)
   Launcher and Missile Status Display (4.2)
   Environmental Display (4.3)
   Engagement Display (4.4)
   Ship Parameter Display (4.5)
   Track Display (4.6)

4. Objects Used by the Module: Display requests.

5. Operations Module Performs: The Display module calls the required modules to generate displays as necessary.

1. Module Name: MENU DISPLAY, NUMBER 4.1

2. Module Purpose: To access the Menu/State Data Base and display the required menu when called and keep track of the state of the program.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Menu/State Data Base.

5. Operations Module Performs: The Menu Display module will access the Menu/State Data Base and provide the necessary menu for display when prompted by the Display module. The Menu Display module will also keep track of the state of the program, that is what menus can be displayed given that the state of the program exists now with a current menu.

1. Module Name: LAUNCHER AND MISSILE STATUS DISPLAY, NUMBER 4.2

2. Module Purpose: To access the Launcher and Missile Status Data Base and provide a display of the information contained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Launcher and Missile Status Data Base.

5. Operations Module Performs: The Launcher and Missile Status Display module will display the information contained in the Launcher and Missile Status Data Base when prompted by the Display module.

1. Module Name: ENVIRONMENTAL DISPLAY, NUMBER 4.3

2. Module Purpose: To access the Environmental Data Base and provide a display of the information contained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Environmental Data Base.

5. Operations Module Performs: The Environmental Display module will display the information contained in the Environmental Data Base when prompted by the Display module.

1. Module Name: ENGAGEMENT DISPLAY, NUMBER 4.4

2. Module Purpose: To graphically display the flight path of missiles that are to be flown against a set target. Threat data on the target will also be displayed. The engagement plan will have the capability to be superimposed over the general track display.

3. Subordinate Modules: Threat Display (4.4.1)
Automatic Engagement (4.4.2)
Manual Engagement (4.4.3)

4. Objects Used by the Module: Threat Data Base information.
Engagement Plan Data Base information.
Manual inputs for an engagement plan.

5. Operations Module Performs: The Engagement Display module calls upon subordinate modules to provide the operator a display of the computer generated engagement plan or a manually generated engagement plan constructed by the operator. In both cases the threat data pertinent to the displayed target can also be shown.

1. Module Name: THREAT DISPLAY, NUMBER 4.4.1

2. Module Purpose: To access the Threat Data Base and provide a display of the information contained in that data base.

3. Subordinate modules: None

4. Objects Used by the Module: The information contained in the Threat Data Base.

5. Operations Module Performs: The Threat Display module will display the information contained in the Threat Data Base when prompted by the Engagement Display module.

1. Module Name: AUTOMATIC ENGAGEMENT DISPLAY, NUMBER 4.4.2

2. Module Purpose: To graphically display the engagement plan that was generated by the Plan Engagement module and stored in the Engagement Plan Data Base.

3. Subordinate Modules: Graphics Display (4.4.2.1)

4. Objects Used by the Module: Information contained in the Engagement Plan Data Base.

5. Operations Module Performs: The Automatic Engagement Display module provides a graphical representation of the engagement plan as contained in the Engagement Plan Data Base. All missile trajectories and waypoints will be depicted with associated missile fire times and arrive over target time. The uncertainty ellipses will also be generated along with the probability of acquisition of the target.

1. Module Name: GRAPHICS DISPLAY, NUMBER 4.4.2.1

2. Module Purpose: To provide graphics capabilities to the Automatic Engagement Display module and the Manual Engagement Display module.

3. Subordinate Modules: None

4. Objects Used by the Module: Engagement Plan Data Base information.
Manually input engagement plan.

5. Operations Module Performs: The Graphics Display module provides the graphics capabilities necessary to the Automatic Manual Engagement Display module to accurately portray their given engagement plans.

1. Module Name: MANUAL ENGAGEMENT DISPLAY, NUMBER 4.4.3

2. Module Purpose: To provide the operator the capability to manually input an engagement plan for attacking a given target.

3. Subordinate Modules: Graphics Display (4.4.2.1)

4. Objects Used by the Module: Information contained in the Threat Data Base.

5. Operations Module Performs: The Manual Engagement Display module allows the operator to manually input his own engagement plan for a given target. Once this information is graphically input to the display, it can be transferred to the Engagement Plan Data Base where it can be programmed to the missiles like an automatically generated plan.

1. Module Name: SHIP PARAMETERS DISPLAY, NUMBER 4.5

2. Module Purpose: To access the Ship Parameter Data Base and provide a display of the information contained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Ship Parameter Data Base.

5. Operations Module Performs: The Ship Parameter Display Module will display the information contained in the Ship Parameter Data Base when prompted by the Display Module.

1. Module Name: TRACK DISPLAY, NUMBER 4.6

2. Module Purpose: To access the Track Data Base and provide a continuous display of all tracks being maintained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Track Data Base.

5. Operations Module Performs: The Track Display module will continuously display all tracks maintained in the Track Data Base. These tracks will be constantly updated as the Track Data Base is updated. The symbology and method presentation of the tracks should closely coincide with NTDS displays.

APPENDIX D

ACRONYMS AND ABBREVIATIONS

BIT - Built-In Test

BITE - Built-In Test Equipment

BOL - Bearing Only Launch

BRG - Bearing

BSTR - Booster

C&C - Command and Control

CP - Casualty Panel

CIC - Combat Information Center

DCU - Data Conversion Unit

DPC - Data Processor Computer

EAS - Engagement Analysis System

EPS - Engagement Planning System

FCS - Fire Control System

HSCLCS - HARPOON Ship Command-Launch Control Set

HWS - HARPOON Weapons System

NTDS - Naval Tactical Data Systems

RNSH - Royal Navy Sublaunched HARPOON

RBL - Range Bearing Launch

TDS - Tactical Data System

UBFCS - Underwater Battery Fire Control System

WCIP - Weapon Control Indicator Panel

WCS- Weapon Control System

APPENDIX E

SYSTEM DESIGN USING ADA

```
Package Update is
    Task  Launcher-Missile_Status is
          entry Update (Launcher-Missile Status:   in Status
            Type);
    End   Launcher-Missile_Status;
    Task  Ship-Parameter is
          entry Update (Ship-Parameter:   in Ship-Parameter-
            Type);
    End   Ship-Parameter;
    Task  Environment is
          entry Update (Environment:  in Environment-Type);
    End   Environment;
    Task  Threat is
          entry Update (Threat:  in Threat-Type);
    End   Threat;
    Task  Update-Track is
          entry Add (Track:  in Track-Type);
          entry Delete (Track:  in Track-Type);
          entry Modify (Track:  in Track-Type);
    End   Update Track;
End Update;
```

```
Package Auto-Engagement is
    Procedure   A-Engagement (Launcher-Missile-Status:  in
                  Status-Type, Threat:  in Threat-Type,
                  Engagement-Plan: out Engagement-Plan-Type);
    Procedure   Prob-of-Acquisition (Engagement-Plan:  in out
                  Engagement-Plan-Type);
    Procedure   Uncertainty-Ellipse (Engagement-Plan:  in out
                  Engagement-Plan-Type);
End Auto-Engagement;


Package Manual-Engagement is
    Procedure   M-Engagement (Launcher-Missile Status:  in
                  Status-Type, Engagement-Plan:  out Engage-
                  ment-Plan-Type);
End Manual-Engagement;
```

```
Package Display is

    Task  Menu-Display is
          entry Access (Menu:  out Menu-Type);
    End   Menu-Display;

    Task  Launcher-Missile-Status is
          entry Access (Launcher-Missile-Status:  out Status-
            Type);
    End   Launcher-Missile-Status;

    Task  Environment is
          entry Access (Environment: out Environment-Type);
    End   Environment;

    Task  Ship-Parameter is
          entry Access (Ship-Parameter:  out Ship-Parameter-
            Type);
    End   Ship-Parameter;

    Task  Track is
          entry Access (Track:  out Track-Type);
    End   Track;

    Task  Threat is
          entry Access (Threat:  out Threat-Type);
    End   Threat;

    Task  Engagement-Plan is
          entry Access (Engagement-Plan:  out Engagement-
            Plan-Type);
    End   Engagement-Plan;


    Package  Engagement-Display is

          Procedure  Manual-Engage-Display (Engagement-Plan:
                     in out Engagement-Plan-Type, Threat:  in
                     out Threat-Type);

          Procedure  Manual-Engage-Display (Engagement-Plan:
                     in out Engagement-Plan-Type, Threat:  in
                     out Threat-Type);

          Procedure Graphics (Engagement-Plan:  in out
                     Engagement-Plan-Type);

    End   Engagement Display;

End Display;
```

```
Package Data Base Managers is
    Package Launcher Missile Status Manager is
        Type Status Type is
            Record
                Empty    :  Boolean;
                Miss type:  String range A .. C;
            End record;
        Task Launcher Missile Status is
            Entry Update (Launcher Missile Status in Status
Type);
            Entry Access (Launcher Missile Status out Status
Type);
        End Launcher Missile Status;
    End Launcher Missile Status Manager;
    Package Ship Parameter Manager is
        Type Ship Parameter Type is
            Record
                Course       :  Integer range 0..359;
                Speed        :  Integer range 0..50;
                Position Lat :  Latitude;
                Position Long:  Longitude;
            End record;
        Task Ship Parameter is
            Entry Update (Ship Parameter:  in Ship Parameter
Type);
```

```
            Entry Access (Ship Parameter:  out Ship Parameter
Type);
        End Ship Parameter;
    End Ship Parameter Manager;
    Package Environment Manager is
        Type Environment Type is
            Record
                Visibility :  Real range 0..30;
                Sea-State  :  Integer range 0..5;
                Wind Dir   :  Integer range 0..359;
                Wind spd   :  Integer range 0..100;
                Temperature:  Integer range -100..150;
                .
                .
                .
            End record;
        Task Environment is
            Entry Update (Environment: in Environment Type);
            Entry Access (Environment: out Environment Type);
        End Environment;
    End Environment Manager;
    Package Threat Manager is
        Type Threat Type is
            Record
                Ship name  :  String;
                Ship class :  String;
                Weapons    :  String;
```

```
            ECM Equip   :  String;
            Attack Plan:  String:
            .
            .
            .
        End Record;
    Task Threat is
        Entry Update (Threat: in Threat Type);
        Entry Access (Threat: out Threat Type);
    End Threat;
End Threat Manager;
Package Track Manager is
    Type Track is
        Record
            Type Track    :  Boolean;
            Class Vessel  :  String;
            Bearing       :  Integer range 0..359;
            Range         :  Integer range 0..500;
            Position Lat  :  Latitude;
            Position Long:  Longitude;
            Course        :  Integer range 0..359;
            Speed         :   Integer range 0..50;
        End Record;
    Task Track is
        Entry Add (Track: in Track Type);
        Entry Delete (Track: in Track Type);
```

```
        Entry Modify (Track: in Track Type);
        Entry Access (Track: out Track Type);
    End Track;
End Track Manager;
Package Menu Manager is
    Type Menu is
        Record
            Undetermined at this time
        End Record;
    Task Menu Display is
        Entry Access (Menu: out Menu Type);
    End Menu Display;
End Menu Manager;
Package Engagement Plan Manager is
    Type Engagement Plan is
        Record
            Track Desig :  String;
            Type Plan   :  Boolean;
            Num Missiles:  Integer range 0..24
            Sequence    :  Array;
            Miss Type   :  String Range A..C;
            .
            .
            .
        End Record;
```

```
        Task Engagement Plan is
            Entry Access (Engagement Plan:  out Engagement
Plan Type);
        End Engagement Plan;
    End Engagement Plan Manager;
End Data Base Manager;
```

LIST OF REFERENCES

1. Booch, Grady, Software Engineering with Ada, The Benjamin/Cummings Publishing Company, Inc., 1983, p. 40-44.

2. Comptroller General of the United States Letter to Congressman Jack Brooks, Subject "DOD Instruction 5000.5X, Standard Instruction Set Architectures for Embedded Computers (MASAD-82-16)", 1982, p. 8.

3. Richardson, Gary L., Butler, Charles W. and Tomlinson, John D., A Primer on Structured Program Design, Petrocelli Books, 1980, p. 3.

4. Maroney, Randal and Sentman, Lawrence, Integrated Design Specifications for the Harpoon Shipboard Command Launch Control Set, Master's Thesis, Naval Postgraduate School, Monterey, California, December, 1982.

5. Softech, Ada Programming Design Language Survey Final Report, October, 1982, pp. 3.18-3.25.

6. Pressman, Roger S., Software Engineering: A Practitioner's Approach, McGraw-Hill Book Company, 1982.

BIBLIOGRAPHY

Barnes, J. G. P., Programming in Ada, Addison-Wesley Publishing Company, 1982.

Freedman, Roy S., Programming Concepts with the Ada Language, Petrocelli Books, Inc., 1982

Wheeler, Thomas J., Embedded System Design with Ada as the System Design Language, The Journal of Systems and Software 2, 11-21 (1981).

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Department Chairman, Code 54 Department of Administrative Science Naval Postgraduate School Monterey, California 93940	1
5.	LT Daniel P. Olivier 515 Zest Court San Diego, California 92134	4
6.	LT Kevin R. Olsen 2910 Charles St. Racine, Wisconsin 53402	4
7.	NSWSES Code 4613 (Mr. Paul Gognon) Port Heuneme, California 93043	1
8.	Naval Sea Systems Command Department of the Navy ATTN: Mr. Lee Minin NAVSEA Code SEA-62WB Washington, D.C. 20362	1
9.	Mr. Fred S. Gais Director - HARPOON Ship Integration McDonnell Douglas Astronautics Company St. Louis Division P.O. Box 516 St. Louis, Missouri 63166	1

10. LCDR Ronald Modes, USN, Code 52MF 4
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940

11. Adjunct Professor William J. Haga, Code 54Hj 1
Department of Administrative Science
Naval Postgraduate School
Monterey, California 93940

12. Naval Postgraduate School 1
Computer Technologies Curricular Office
Code 37
Monterey, California 93940

201626

Thesis
046 Olivier
c.1 A design methodology for embedded weapons systems using the Harpoon shipboard command-launch control set (HSCLCS), AN/SWG-1A (V).